MASTER DISSERATION

2023-2024

Towards Automated Complexity Grading: A Python-based Natural Language Processing Application for Textual Analysis of Japanese

MASTER IN TRADUMATICS: TRANSLATION TECHNOLOGIES

FACULTY OF TRANSLATION AND INTERPRETATION

Author: Javier Sánchez Jiménez

NIU: 1696436

TUTOR

Antoni Oliver Gonzàlez

Barcelona, 17th of July 2024

Dissertation data / *Datos del TFM*

Title: Towards Automated Complexity Grading: A Python-based Natural Language Processing Application for Textual Analysis of Japanese

Título: Evaluación automática de la complejidad: una aplicación de procesamiento del lenguaje natural basada en Python para el análisis textual del japonés

Author: Javier Sánchez Jiménez

Autor: Javier Sánchez Jiménez

Tutor: Antoni Oliver Gonzàlez

Tutor: Antoni Oliver Gonzàlez

Centre: Autonomous University of Barcelona (UAB)

Centro: Universidad Autónoma de Barcelona (UAB)

Studies: Official master's degree in Tradumatics: Translation Technologies

Estudios: Máster oficial en Tradumática: Tecnologías de la traducción

Keywords / *Palabras clave*

Natural Language Processing, Japanese, Textual Analysis, Python

Procesamiento del lenguaje natural, japonés, análisis textual, Python

Abstract / *Resumen del TFM*

Abstract….

The Japanese language presents numerous peculiarities that make it an intriguing and challenging subject for Natural Language Processing (NLP). These include the use of three different writing systems, the absence of spaces between words, and its nature as an agglutinative language.

Combined with a lack of non-Japanese bibliography on the subject, these characteristics offer a unique opportunity for deeper exploration into the NLP field. This project aims to address these challenges through the development of a Python-based application designed to grade Japanese written texts within the JLPT framework by analysing various textual features.

Resumen…

El idioma japonés presenta numerosas peculiaridades que lo convierten en un sujeto intrigante y desafiante para el procesamiento del lenguaje natural (PLN). Estas incluyen el uso de tres sistemas de escritura diferentes, la ausencia de espacios entre las palabras y su naturaleza como lengua aglutinante.

Combinadas con la escasez de literatura en otros idiomas que no sea en japonés, estas características ofrecen una oportunidad única para una exploración más profunda en el campo del PLN. Este proyecto tiene como objetivo abordar estos desafíos mediante el desarrollo de una aplicación basada en Python diseñada para calificar textos escritos en japonés dentro del marco JLPT mediante el análisis de varias características textuales.

Legal notice / *Aviso legal*

Dedicado a mis padres por siempre confiar en mí, y a mi hermano, porque sin él este proyecto nunca hubiera sido posible.

INDEX

INDEX OF TABLES AND FIGURES

All the source files written for the development of this application can be found at: https://github.com/milovatjp/hazuki

1. Introduction

This master's dissertation will focus on the development of an application capable of analysing written texts in Japanese within the framework of the Japanese Language Proficiency Test (JLPT) using various Python libraries for Natural Language Processing. To achieve this, different existing texts classified by proficiency level and specifically created for Japanese language learning will be processed, along with vocabulary lists, grammatical structures, and *kanji* characters. The textual features analysed are sentence length in characters, syntactical units per sentence, proficiency level of the vocabulary, *kanji-kana* ratio of the characters present in the text, and *kanji-kana* ratio of the words present in the text.

1.1 Motivation

Understanding how Natural Language Processing works is one of my main goals when studying this master's degree. I am really interested in it, and I believe it is a great opportunity for applying all the theoretical concepts learned about this topic. Additionally, I am passionate about Japanese language and, in comparison to other languages (especially European languages), the progress made in NLP for Japanese is not as advanced.

Developing an application for analysing and classifying textual complexity will allow me to work directly with tools used in the computational linguistics field, improve my coding skills, and gaining a deep understanding of NLP. It will also allow me to create a potentially useful tool that can be of great value for future Japanese learners. This is a project in I which I would like to keep working on in the future, improving it as I learn more and hopefully publishing it online so others can benefit from it.

1.2 Objectives

The main objective of this project is to develop an application that is useful and accessible for Japanese students who may not have resources or enough proficiency level in Japanese to determine themselves whether the texts they are dealing with are appropriate for their level or not.

A good and easy to understand example of this is *kanji* knowledge. One of the biggest barriers (if not the biggest) for Japanese learners is their inability to recognize *kanjis*. In opposition to graded readings, everyday language does not separate easy or common *kanjis* from complex and rarely used ones, and not understanding one of these may lead the reader to a complete misunderstanding of what is written. One of the most basic features of this application will be to let the user know how many vocabulary words present in the text are unknown to them, based on the user's level within the JLPT reference framework. Additional features of the application are length sentence, syntactical units per sentence, and vocabulary classification by level of JLPT proficiency.

The textual features used for this analysis are length in syntactical units, length in characters, number of *kanji* characters in the text, number of *kana* characters in the text, proficiency level of words and grammatical category of words, among others.

Texts may be easy sentences, paragraphs, news articles, novels, or subtitles. Whatever the learners are interested consuming, this application will give them detailed textual information that may prevent them from wasting their time with contents that are way above their proficiency level.

2. Literature Review

2.1 General Overview of Natural Language Processing

Natural language processing, often shortened as NLP, is an interdisciplinary field of linguistics which revolves around the relationship between human language and computers. Manris (1999) defines it as:

> *NLP could be defined as the discipline that studies the linguistics aspects of human-human and human-machine communication, develops models of linguistic competence and performance, employs computational frameworks to implement process incorporation such models, identifies methodologies for iterative refinement of such processes/models, and investigates techniques for evaluating the results.*

Feng (2023) divides the processing of the natural language by computers in four aspects, which are formalization, algorithmzation, programming, and practicalization. Feng (2023) states that linguistic problems should be formalized in a mathematical form, then this regular mathematical form should be expressed in terms of an algorithm. Following this, a computer program based on this algorithm should be written to formulate different NLP systems, and finally, the established NLP system should be continuously evaluated in order to improve its quality and performance.

The establishment of NLP models require nine specifications of linguistic knowledge, which are: acoustics, phonology, morphology, lexicology, syntax, semantics, discourse, pragmatics, and common-sense encyclopaedia knowledge (Feng, 2023). Based on these nine specifications, Feng (2023) addresses that NLP as a linguistic problem, however, he states that NLP might also involve other knowledge from areas such as: computer science, mathematics, psychology, philosophy, statistics, electronic engineering, and biology.

The process of NLP has been traditionally divided into stages which aim to mirror the theoretical linguistics distinctions between syntax, semantics, and pragmatics (Dale, 2009). Based on this distinction, the NLP general process would appear as:

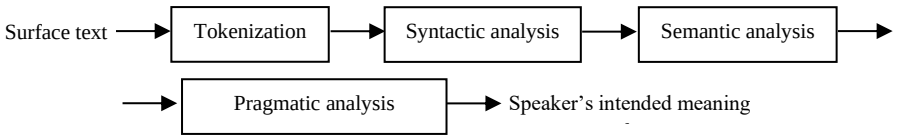


Figure 1: Natural Language Process

Tokenization is the first step in this NLP, and it involves splitting text into basic units that hold meaning. These units are called tokens and, depending on the approach, they can be words, sentences, characters, or complete phrases.

Word-based tokenization is the most common approach, splitting text at whitespace to create individual word tokens. This approach is used for Indo-European languages as the existence of whitespaces as word delimiters facilitates the task of tokenization. In languages

where the delimitators between words are not explicitly displayed and lack the usage of whitespaces, such as Chinese, Japanese, or Thai, this approach is not valid.

Sentence-based tokenization approach relies on punctuation marks to determine the existence of sentences within a text stream. This approach is useful for sentiment analysis and summarization. Successful sentence segmentation for a given language requires a deep understanding of the punctuation characters in that language. Written languages that do not make strict use of punctuation marks present a problem for this kind of tokenization (Palmer, 2009). Additionally, texts with poor punctuation, or situations in which a punctuation character might have different functions, present a problem for delimiting sentence boundaries.

Character-based tokenization approach breaks the text stream down to individual characters, often used for machine translation and named entity recognition (NER).

N-gram tokenization approach creates sequences of n consecutive words to capture word co-occurrence patterns.

Lexical analysis examines each token to determine its lexical properties. This process provides a deeper understanding of the individual words within a sentence and enables tasks like sentiment analysis (positive vs. negative words) or dependency parsing, understanding relationships between words. Its components are:

- Lexicon (Dictionary): Stores information about words, including part-of-speech tags, synonyms, and definitions.
- Morphology analysers: Break down complex words into their root forms (stems) or base words (lemmas). This helps identify variations of the same words (e.g., “wark”, “walking”, “walked”).
- Part-of-speech (POS) taggers: Assign a grammatical category (noun, verb, adjective, etc.) to each token.

Syntactical analysis is the process of analysing the grammatical structure of a sentence by examining the relationships between tokens. This analysis enables tasks like machine translation or question answering, as understanding of sentence structure is necessary for accurate translation and for identifying key elements in a question. To do so two main techniques exist:

- Phrase structure grammar: Representation of sentences through hierarchical tree structures, providing visual representation of how phrases combine to form sentences.
- Dependency parsing: Identifies the grammatical dependencies between words, indicating how each word relates to another (e.g., subject-verb, verb-object).

Pragmatic analysis is the final step in this process, and it goes beyond grammar to understand the speaker’s or writer’s intent and the context surrounding the utterance.

2.2 Challenges in Natural Language Processing for Japanese

When compared with other languages such as English, the development of NLP Japanese is not only less advanced but faces different kind of technical problems. The first and most obvious is tokenization and segmentation. Japanese language does not have spaces between

words, making tokenization and segmentation more complex. Traditional approaches, such as using whitespaces or punctuation as token boundaries are less effective. The lack of spaces between words necessitates a more informed approach than simple lexical analysis, which will vary depending on the language (Palmer, 2009). In the case of Japanese, morphological analysis and dictionary-based tokenization are approaches commonly used to segment text into meaningful units, such as morphemes or characters.

The Japanese writing system combines alphabetic, syllabic, and logographic symbols, and modern Japanese texts make use of *kanji* (Chinese *hanzi* symbols), *hiragana* (syllabary for grammatical particles and words of Japanese origin), *katakana* (syllabary for words of foreign origin), *romaji* words (words written in the Roman alphabet), Arabic numerals, and punctuation marks. This abundance in character sets makes it easier to tokenize Japanese, but still does not make it for the lack of segmentation between words. Additionally, the change of writing system does not always mean the end and beginning of a new word, as it is very common for words to combine multiple of these characters sets, such as inflected verbs, which often combine *kanji* for the root and *hiragana* for the inflectional ending.

Additionally, Japanese is classified as an agglutinative language, meaning that it forms words and grammatical constructions by adding affixes to a base or root form. This becomes an extra challenge to the aforementioned and omnipresent problem of not having explicit word segmentation. As an agglutinative language, Japanese word formation is heavily based on the addition of suffixes, and this makes is hard for NLP systems to accurately segment words and identify their constituent morphemes. This agglutinative nature gets furtherly complicated by the fact that single morphemes may have multiple meanings or functions depending on its context and the affixes attached to it, leading to ambiguity in word segmentation and morphological analysis. A Japanese search engine must be capable of segmenting the search term into meaningful semantic units while being capable of ignoring morphological variants like the conjugations (Halpern, 2007).

This is achieved through the use of conflation, which refers to merging multiple forms of a word into on representation. In Japanese, conflation involves treating different forms of words as equivalents. For example, merging 食べます (*tabemasu*, polite present tense), 食べる (*taberu*, present tense), and 食べた (*tabeta*, past tense) into a single representation for analysis or processing, recognizing them all as variations of the verb “to eat”. Within this kind of NLP techniques exists the process of stemming and lemmatization. On one hand, stemming removes the suffixes from words to reduce them to their root form. This is not always valid approach as the resulting stem might not always be a valid word. On the other hand, lemmatization reduces words to their base dictionary form, known as a lemma.

In the introduction of its publication “The Complexities of Japanese Homophones”, Jack Halpern used the following words to define Japanese orthography:

> *Japanese orthography is so highly irregular that it can be considered, without the slightest fear of being accused of hyperbole, to be a couple of orders of magnitude more complex and more irregular than any other major language, Chinese included. A major source of complexity in processing Japanese texts is the presence of an extremely large number of* ***homophones.***

Within the introduction of this publication, Halpern sheds some light in this topic with an eye-opening example that can help those who are not knowledgeable enough of the

complexities of Japanese language. He provides us with the sentence *Hi no sasanai yashiki (A Mansion with no Sunshine)*, which he describes as the potential name of a novel or a film. Halpern gives us twelve different and legitimate ways of writing that sentence.

1. 日の差さない屋敷
2. 日の射さない屋敷
3. 日のささない屋敷
4. 日の射さない邸
5. 日の差さない邸
6. 日のささない邸
7. 陽の射さない屋敷
8. 陽の差さない屋敷
9. 陽のささない屋敷
10. 陽の射さない邸
11. 陽の差さない邸
12. 陽のささない邸

With this simple example, it is possible to see the difficulties that homophones present for NLP, as text will not always be presented in the most standard form, or in its most "correct" way of writing, because there is not such. The following table contains the complex semantic interrelations between the homophones for *sasu,* lemma for *sasanai* in the previous example sentence: *Hi no sasanai yashiki*.

Homophones for *sasu*				
No.	**English**	**"Standard" Form**	**Sometimes also**	**Often also**
1	to offer	差す		さす
2	to hold up	差す		さす
3	to pour into	差す	注す	さす
4	to color	差す	注す	さす
5	to shine on	差す	射す	さす
6	to aim at	指す	差す	
7	to point to	指す	さす	
8	to stab	刺す	さす	
9	to leave unfinished	さす	止す	

Figure 2: Homophones for sasu. Jack Halpern, The CJK Dictionary Institute, Inc.

Halpern (2007) considers that retrieving or identifying Japanese homophones is not, in itself, more complicated than doing it for English, for example *write* and *right.* Halpern summarizes in three the characteristics present in Japanese homophones that present difficulties in Japanese text processing:

- Homophones are nearly synonymous or even identical in meaning. There is no way to predict which particular homophone will appear in a text.
- The distinction between some homophones is so subtle that many authors just decide to sidestep the problem of selecting the appropriate *kanji* and resort to *hiragana*.
- Japanese language has a small stock of phonemes, hence, the number of homophones is very large.

Halpern argues that merely retrieving all homophones in processing technology does more harm than good since it matches numerous irrelevant homophones, such as 帰る *kaeru* "to return" and 変える *kaeru* "to change" and gives an insight into the fact that homophone processing techniques require comprehensive database of semantically and etymologically classified homophones.

In the same vein of homophones processing, homographs disambiguation presents a real challenge for Japanese NLP. According to the CJKI (Chinese, Japanese, Korean Institute), over 20.000 homographs are present within their databases, and it is self-evident that, since homographs are written the same way, it's a complicated task to retrieve the semantical relevant one during a search. Halpern gives us some of the most typical examples on the multiple readings of kanji:

Japanese Homographs			
No.	**Homograph**	**Reading**	**English**
1	一時	ichiji	one o'clock; temporarily
	一時	hitotoki	a while
	一時	ittoki	a moment; 12th part of day
2	一章	isshoo	one chapter
	一章	kazuaki	a first name
3	仮名	kana	kana syllabary
	仮名	kamei	fictious name, pseudonym
	仮名	karina	alias, assumed name
	仮名	kemyoo	fictious name, pseudonym
4	化学	kagaku	chemistry
	化学	bakegaku	chemistry

Figure 3: Japanese Homographs Jack Halpern, The CJK Dictionary Institute, Inc.

2.3 Progress in Natural Language Processing for Japanese

Within the NLP field, the emerging LLM (Large Language Models) have meant a change in how NLP is conceived. LLM are neural network-based models designed to understand and generate human-like text. These models are trained on large amount of text data and can be fine-tuned for specific tasks such as language translation, text summarization, or question answering, among others. The two most know models are GPT (Generative-Pre trained Transformer) and BERT (Bidirectional Encoder Representations from Transformers).

In the case of Japanese LLM, most of the existing LLM are built upon some of the variants of the GPT or Llama architectures or built off some of theirs existing models. The most relevant LLMs focused on Japanese language are:

- OpenCALM, a GPT-NeoX architecture LLM developed by CyberAgent
- Stormy, a GPT-NeoX architecture LLM developed by the University of Tokyo

- rinnaGPT, a GPT-NeoX architecture LLM developed by rinna

The performance of these LLM was judged based on LLM Judge model developed by Lianmin Zheng, Wei-Lin Chiang and Ying Sheng (2023). A Japanese fine-tuned version of this model called Rakuda benchmark was used to compare the performance of these three Japanese LLM and GPT-3, and then compared against the results of GPT-4 to act as judge and asses the quality of the models' responses.

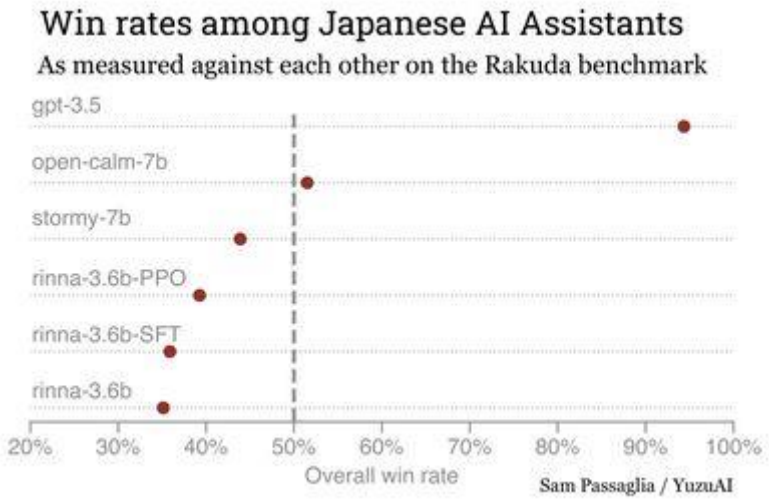


Figure 4: Win rates among Japanese AI Assistants.Lianmin Zheng, Wei-Lin Chiang and Ying Sheng (2023)

In comparison with English language, the existing number of LLM is considerably small. Additionally, the availability of training data for Japanese is very limited. The following graph, taken from the LLM-jp project's GitHub, carried out by Kaito Sugimito (2023), compares the number of parameters used by GPT of Japanese and English since May 2020 until today.

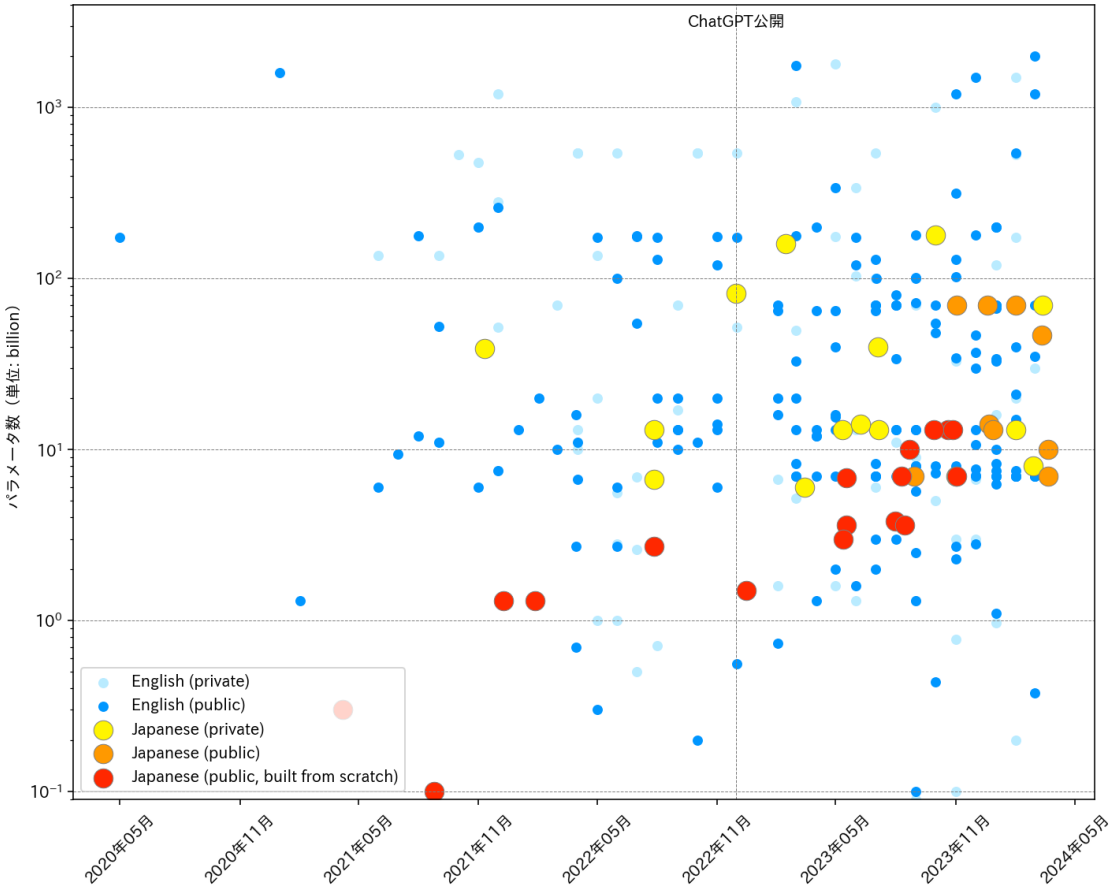


Figure 5: Evolution of parameter sizes for Japanese LLMs and English LLMs. LLM-jp.

Japanese Text Generation Models used for general purpose have been trained from the same training data, both the models created from scratch and the models built off English LLMs. The most common pre-training sets often are Japanese Wikipedia, Japanese mC4, Japanese CC-100, and Japanese OSCAR.

For more detailed information of parameters, base models, training data, developers, or licenses, check the LLM-jp GitHub repository.

3. Theoretical Framework

3.1 Key Concepts of Japanese Linguistics

3.1.1 Japanese as an Agglutinative Language

Japanese is an agglutinative language, which means it forms words and expresses grammatical relationships through the addition of affixes to a base or root word. This characteristic is evident in its use of particles, verb conjugations, and the incorporation of various suffixes to indicate tense, mood, politeness, and other grammatical nuances. Unlike inflectional languages, where a single morpheme might convey multiple pieces of grammatical information, or isolating languages, where words remain unchanged, Japanese builds complex meanings by sequentially attaching morphemes. This structure allows for a high degree of precision and subtlety in communication.

3.1.2 Bunsetsu: Phrasal Units

Bunsetsu (文節) are fundamental syntactic units in Japanese language, typically consisting of a content word such as a noun, verb or adjective followed by zero or more function words such as particles or auxiliary verbs. They serve building blocks for sentence, each representing a basic phrase that can stand alone in term of meaning and grammatical function.

Zhang and Ozeki (1998) argued that the fact that there are no spaces to indicate *bunsetsu* boundaries in the orthographic writing of Japanese, therefore, a sentence must be segmented into *busetsus* somehow prior to dependency structure analysis, and that these segmentations have been traditionally performed by using hand-crafted rules.

Bunsetsu has been a complex scope of study in the field of NLP, with current studies on parsing applications, algorithms and statistical-based learning methods to support *bunsetsu*-based dependency parsing (Butler et. al, 2012). For further information, the paper "Problems for successful *bunsetsu*-based parsing and some solutions" by Butler et. al 2012 shed some light on the basics of this field. Establishing efficient NLP methods capable of distinguishing and learning the dependency relations between content words and function words through dependency parsing is essential. These methods should understand grammatical patterns and capture dependency structures, relying on techniques such as POS tagging to perform *bunsetsu* parsing.

For the purposes of this dissertation, it is needed to simply understand the concept of *bunsetsu* and how, parsing a given text into these syntactic units, can help a student to break long and complex sentences into simpler chunks that can be easily processed and understood.

3.2 Japanese Writing Systems

One of the uniqueness of Japanese language is the simultaneous use of three different native writing systems, which are *hiragana*, *katakana*, and *kanji*. All three systems can be used together in a single sentence, providing a visually complex but information-rich text. As

Japanese language does not make use of whitespaces or any other kind of explicit delimitator between words, the coexistence of these three writing systems aids creating a natural way of differentiating words, but it is not always a reliable solution.

3.2.1 Hiragana and Katakana

Hiragana and *katakana* are two phonetic syllabaries, meaning each character represents a single sound (consonant-vowel combination).

Hiragana is primarily used for:

- Grammatical elements like particles (の (*no*) for "of")
- Auxiliary verbs (ている (*teiru*) for "being")
- Verb and adjective conjugations (e.g., 食べる (*taberu*) "to eat," 食べます (*tabemasu*) "eats")
- Native Japanese words that don't have corresponding *kanji* (e.g., こんにちは (*konnichiwa*) "hello")

The syllabary consists of 46 basic characters, with additional combinations not directly represented (voiced consonants and vowel length). Visually speaking, *hiragana* characters are known for their rounded, flowing strokes, often described as more cursive.

Figure 6: Hiragana chart

Similar to *hiragana, katakana* is also a phonetic syllabary with 46 characters. It is primarily uses are:

- Foreign words (e.g., アメリカ (*amerika*) for “America)

- Loanwords from other languages and adapted to Japanese (e.g., コンピューター (*konpyūtā*) for "computer")
- Names. Specially for people and places not originally written in *kanji* (e.g., アンナ (*anna*) for the name "Anna")
- Onomatopoeia. Words that imitate sounds (e.g., バタン (*batan*) for "slam")
- Scientific terms, often written in *katakana* for clarity (e.g., ディー・エヌ・エー (*dī-en-e*) for “DNA”)
- Emphasis of words or phrases

Katakana characters are angular, with blocky strokes, and a stricter appearance than *hiragana.*

Figure 7: Katakana chart

3.2.2 Kanji

Kanji characters have their origin in Chinese characters, *hanzi*, adopted around the 5th century AD. They are ideograms and represent words or concepts, often with multiple readings and meanings depending on context. Their readings are divided into two differentiated categories: semantic reading and phonetic reading.

- Semantic reading or meaning-based reading are those in which the character conveys the meaning directly (eg., 水 (*mizu*) for “water”).
- Phonetic reading or sound-based reading are those in which the character is used for its sound, often corresponding to a related Chinese word (e.g., 水 (*mizu*) in “水泳 (*suiei*)” for “swimming”).

Kanji are primarily used for nouns, verbs, and adjectives, taking the core position of words, and in some situations, they can also be used for adverbs and conjunctions.

There are thousands of *kanji* characters, with some estimates reaching over 50,000. However, a working knowledge of a few thousand is sufficient for basic literacy. Since the end of World War II, the Japanese Ministry of Education has been working on a list of *kanji* which are considered a literacy baseline. The latest update of this list is called “regular-use *kanji*” (常用漢字), and it was published in 2010. It consists of 2,136 *kanji*: 1,026 *kanji* taught in primary school and 1,110 *kanji* taught in secondary school. All official government documents are restricted to the usage of these *kanjis.*

3.3 Japanese Language Proficiency Test (JLPT) as a Reference Framework

Natural language differs from language imposed in classrooms by the mere fact that the latter seeks to categorize language into distinct difficulty stages for easier learning. Traditional methods aim to dissect language into discrete elements like grammar rules, vocabulary lists, and pre-defined "easy" or "advanced" patterns. However, these constructed frameworks are absent in natural language use. Everyday speech interweaves elements traditionally categorized as easy, intermediate, and advanced by language instruction. Tenses, grammatical cases, conditional structures, and diverse topics co-exist in natural conversations, regardless of their perceived complexity by standardized learning methods.

Despite language being in constant change, evolution, and adaptation, we need to artificially dissect it into a defined framework in order to study it, and this has been traditionally considered the best and only approach to human language learning. Krashen (1989) defied this approach in his book Principles and Practice in Second Language Acquisition, arguing that we humans learn our second language through comprehensible input, by focusing on content and not in grammar, and highlighting that input needs to be engaging for the learner. He advocates for an immersion approach where understanding language takes priority over formal learning, grammar understanding, and vocabulary memorization.

These principles do align with the way humans learn and understand language, however, in the field of NLP, computers need complex language models with predefined patterns, exhaustively defined grammar rules, and extensive list of vocabulary for understanding language. It is due to this reason, that the development of this application needed of a solid foundation and a predefined framework on Japanese language.

The JLPT, which stands for Japanese-Language Proficiency Test, is a standardized test designed to assess the Japanese ability of non-native speakers. It evaluates the student’s knowledge of the language, along with its reading and listening comprehension skills. The JLPT consists of five levels, N5 being the most basic and N1 being the most advanced. Based on the official JLPT website, a brief summary of the linguistic competences of each level would be as it follows:

- N5 (Beginner): Understand and use basic Japanese for everyday situations. Can recognize *hiragana* and *katakana* characters, along with basic *kanji*.
- N4 (Elementary): Comprehend frequently encountered Japanese in everyday settings. Able to read short, simple texts and understand conversations on familiar topics.
- N3 (Intermediate): Grasp the main ideas of everyday Japanese on both familiar and unfamiliar topics. Can read newspapers and articles with moderate complexity.

- N2 (Upper Intermediate): Understand the essential points of complex information on both concrete and abstract topics. Able to read a variety of written materials, including technical documents.
- N1 (Advanced): Comprehend a wide range of demanding, longer texts, and grasp implicit meaning. Can express oneself fluently and spontaneously in complex situations.

In addition to these competencies, it is mentioned that language knowledge, such as vocabulary and grammar is also required for successful execution of the activities.

Based on this framework and supported by extensive *kanji* and vocabulary lists for each level, it was possible to create a solid foundation of data to determine the degree of complexity of a given text.

4. Methodology

4.1 Data Collection

4.1.1 Selection of Texts

The texts were extracted from the Japanese Non-Profit Organization *Tadoku*. These texts were handmade crafted with the aim of create graded reading materials for Japanese language learners, and they are classified into six different levels of difficulty. This dissertation makes use of 848 pages of level 0 texts, 325 pages of level 1 texts, 252 pages of level 2 texts, 216 pages of level 3 texts, and 155 pages of level 1 texts. A total of 1796 pages of graded texts in Japanese.

According to their official data, these levels were designed taking into consideration the following:

	L0 Starter	**L1 Beginner**	**L2 Upper-Beginner**	**L3 Lower-Intermediate**	**L4 Intermediate**	**L5 Upper-Intermediate or above**
Summary	Starter level. Printed from left to right for starting beginners instead of top to bottom. Look carefully at the pictures and illustrations and the story will become self-evident.	Level 5 of Japanese Language Proficiency Test (JLPT). Same vocabulary and grammar as in Level 0 but stories are longer. The lines are perpendicular in levels 1 to 5.	Level 4 of JLPT. Grammar less controlled than Level 1.	Level 3 of JLPT. Grammar less controlled than Level 2. Contents are more varied with fewer pictures.	Levels 3 and 2 of JLPT. Grammar even less controlled than Level 3. Katakana have no reading printed next to them. Stories are longer with even fewer pictures.	Levels 2 and 1 of JLPT. More abstract vocabulary and idiomatic expressions and longer stories than Level 4. No reading hints for Kanji that Year 2 Japanese children should have learned.
Vocabulary range	350	350	500	800	1,300	2,000
Word count/book	0 ~ 400	400 ~ 1,500	1,500 ~ 3,000	2,500 ~ 6,000	5,000 ~ 15,000	8,000 ~ 25,000
New grammar elements	present form, past form, interrogative, ～たい, etc. ※です and ます endings in the main.	present form, past form, interrogative, ～たい, etc. ※です and ます endings.	dictionary form, て−form, ない-form, nominal modification, ～と (conditional), ～から (cause), ～なる, ～のだ, etc.	potential form, imperative form, ～とき, ～たら・ば・なら, ～そう (appearance), ～よう (conjecture, metaphor), compound verb, etc.	causative form, causative passive form, ～そう (information), ～らしい, ～はず, ～もの, ～ようにする/なる, ～ことにする/なる, etc.	Function words, compound words, idiomatic expressions, honorific expressions such as ～わけにはいかない/～につれて/切り開く/召し上がる/伺う.

Figure 8: NPO Tadoku levels

4.1.2 Kanji Lists

Kanjis are compared against a simplified version of the KANJIDICT Project KANJIDIC2 list. This is a project carried out by the Electronic Dictionary Research and Development Group (EDRDG), which aims to compile comprehensive information on *kanji* used in Japanese text processing.

For the purpose of this dissertation, the list was delimited to only 2200 *kanjis* sorted out by frequency. *Kanjis* appearing at the beginning of the list are far more common that the ones in the middle to the end of the list, hence, a text with multiple uncommon *kanjis* would be inevitably harder for a learner to understand.

4.1.3 Vocabulary Lists

Vocabulary is divided into five different lists, each of them corresponding to a JLPT level. N5 has 669 words, N4 has 634 words, N3 has 1834 words, N2 has 1834 words and N1 has 3476 words. This makes a total of 8447 vocabulary words which tokenized input text will be compared against.

These lists are CSV files that consist of three values, which are the *kanji* writing of the word (if existing), the *kana* writing of the word, and the English equivalent or equivalents. It is worth mentioning the importance that the *kanji* and *kana* fields have, because depending of the topic, target audience, or complexity of the text, the same word could be written in *kanji* or in *kana*, This is a relevant aspect to keep in mind and was considered when developing the application, as the tokenized input text's lemmas are compared against both the *kanji* and the *kana* field to make sure no word is omitted due to their graphic representation.

4.1.4 Dictionaries

The dictionary intended to use is a simplified version of the Electronic Dictionary Research and Development Group (EDRDG) JMdict dictionary. This simplified version was intended to be used instead of the original due to multiple reasons. First, it uses a simpler JSON format rather than the advanced original XML format. Secondly, it has a fixed structure for every entry and every value is explicitly pointed. Finally, the field names are comprehensive for human reading, as it does not use complex abbreviations without explanations.

In this simplified version, each entry represents a word or a phrase, and each entry contains necessary fields that are: ID, *kanji* representation, *kana* representation, English equivalent, part-of-speech, and its level of commonness. It also contains other fields that give additional information such as: related, field, dialect, or language source.

Here is an example of an entry for the word 腹ペコ which is a colloquial way to say "hungry":

```
{
  "id": "2080610",
  "kanji": [
```

```
  {
    "common": false,
    "text": "腹ペコ",
    "tags": []
  },
  {
    "common": false,
    "text": "腹ぺこ",
    "tags": []
  }
],
"kana": [
  {
    "common": false,
    "text": "はらぺコ",
    "tags": [],
    "appliesToKanji": ["腹ペコ"]
  },
  {
    "common": false,
    "text": "はらぺこ",
    "tags": [],
    "appliesToKanji": ["腹ぺこ"]
  },
  {
    "common": false,
    "text": "ハラペコ",
    "tags": [],
    "appliesToKanji": []
  }
],
"sense": [
  {
    "partOfSpeech": ["adj-no", "adj-na"],
    "appliesToKanji": ["*"],
    "appliesToKana": ["*"],
    "related": [["ぺこぺこ", 1]],
    "antonym": [],
    "field": [],
    "dialect": [],
    "misc": [],
    "info": [],
    "languageSource": [],
    "gloss": [
      {
        "lang": "eng",
        "gender": null,
        "type": null,
```

```
          "text": "hungry"
        },
        {
          "lang": "eng",
          "gender": null,
          "type": null,
          "text": "starving"
        }
      ]
    }
  ]
}
```


This simplified version is further simplified including only essential information. Removing several fields from the “sense” section and merging all the entries within the “gloss” section into a single array containing all the translations. The resulting simplified structure would look like this:


```
{
  "id": "2080610",
  "kanji": [
    {
      "text": "腹ペコ",
      "common": false,
      "tags": []
    },
    {
      "text": "腹ぺこ",
      "common": false,
      "tags": []
    }
  ],
  "kana": [
    {
      "text": "はらペコ",
      "common": false,
      "tags": []
    },
    {
      "text": "はらぺこ",
      "common": false,
      "tags": []
    },
    {
      "text": "ハラペコ",
      "common": false,
      "tags": []
    }
```

```
  ],
  "sense": [
    {
      "partOfSpeech": ["adj-no", "adj-na"],
      "related": [["ぺこぺこ", 1]],
      "gloss": [
        "hungry",
        "starving"
      ]
    }
  ]
}
```


Originally, the intention was to include this simplified dictionary into the application, allowing the user to consult essential information about the words in the text that are unknown. However, this feature was not implemented as it was soon realized that the English equivalent of the words were already implemented in the CSV files for the Vocabulary lists. This meant a lack of information in the output given to the user, especially the Part-of-speech information. The main reasoning behind this decision has been the fact that CSV Vocabulary lists are sorted by JLPT level, which for the purposes of this application is essential. Meanwhile, despite giving a more extensive list of words and information, the dictionaries lack a proficiency level-based categorization of the word.

It is planned to implement this in an effective way into the application, potentially adding another field to the CSV Vocabulary lists that include POS information without the need of reading a whole dictionary only for that.

4.2 Python Libraries

The development of this dissertation required the usage of NLP tools that allowed to process large amounts of text and had the ability to understand language.

4.2.1 spaCy

spaCy is an open-source library for advanced natural language processing written in Python. It can be used for information extraction, natural language understanding, and preprocessing of deep learning texts. Its functions are:

- Tokenization: Split text into words and punctuation.
- Part-of-speech (POS) tagging: Assign word types (verbs, nouns, etc.) to tokens.
- Dependency parsing: Assign syntax dependency (relationships between individual tokens) labels to tokens.
- Lemma: Assignment of basic forms of words.
- Sentence Boundary Detection (SBD): Detects sentence boundaries.
- Named Entity Recognition (NER): Labels named entities (real objects) such as people, companies, and locations.

- Entity Linking (EL): Converts an entity into a unique knowledge base identifier.
- Similarity: Word, text span, text comparison, and similarity calculation.
- Text classification: Assign a category (or label) to all or part of the text.
- Rule-based matching: Search for sequences of tokens based on text and language annotations, similar to regular expressions.
- Training: Update and improve predictions in statistical models.
- Serialization: Save the object to a file (or byte string).

In the development of this application, SpaCy is used to load GiNZA's language model "**ja_ginza_electra**"

4.2.2 GiNZA

GiNZA is an open-source Japanese natural language processing library. It is built based on SpaCy's framework and uses the open-source morphological analyser SudachiPy for the tokenization process. Its main functions are sentence boundary analysis, morphological analysis, dependency analysis, named entity extraction and phrase extraction.

GiNZA currently has two main language models available which are **ja_ginza** and **ja_ginza_electra**.

- **ja_ginza** is a language model based on SpaCy's architecture and includes pre-trained word vectors for Japanese. It is designed to achieve an overall good performance on general-purpose NLP tasks such as text segmentation, part-of-speech tagging, named entity recognition and dependency parsing.
- **ja_ginza_electra** language model incorporates ELECTRA (Efficiently Learning an Encoder that Classifies Token Replacements Accurately) architecture, a more recent and powerful language model developed by Google. It achieves a higher accuracy and better performance, specially in complex NLP tasks.

ELECTRA language model architecture training involves replacing some input tokens with incorrect ones and training the model to distinguish between original and replaced ones.

For the development of this application, "ja_ginza_electra" language model, and two of the GiNZA's main functions were used: tokenization and *bunsetsu* parsing. The tokenization process is a simple process of parsing the given text into smaller units called "tokens". Each of these tokens is saved into a JSON file in which different information can be stored. The basic information entries and the one used in this application are text, normalized form, part-of-speech and lemmatized form. GiNZA makes use of morphological analysis to properly parse tokens.

This example shows the entry from a JSON file for the word *yukata*, which is a traditional Japanese summer garment. "Text" field is how the word appears in the input text, "norm" is the normalized form of the word, which in this case would be written in *kanji*, "pos" is Part-

of-Speech, or grammatical category, which in this case is noun, and finally "lemma" is the lemmatized form of the word.

Lemmas are of special interest, as this field was the one used as reference when comparing tokens against vocabulary list and other tokens. POS was also used as further reference when same words had different grammatical categories, using POS field as an aid to remove repetitions in the data frames.

Busetsu parsing function was used to offer the user a way to split sentences into smaller and more comprehensive chunks.

The following example, taken directly from GiNZA's blog, clearly shows the difference between tokenization and *bunsetsu* parsing:

The sentence "赤い車を持っている" is directly translated as "I have a red car". On the left, the sentence is tokenized into six tokens, the smallest unit of meaning possible, while on the right, the sentence is split into three *bunsetsus*. A token is the smallest unit of meaning or syntax in language processing, while a *bunsetsu* is a syntactic unit in Japanese consisting of a content word and its associated function words.

This differentiation can be directly translated to the purposes of this application. Tokens are useful for NLP dependencies to parse the text, while *bunsetsus* are useful to the user to properly parse the meaning present in the sentences.

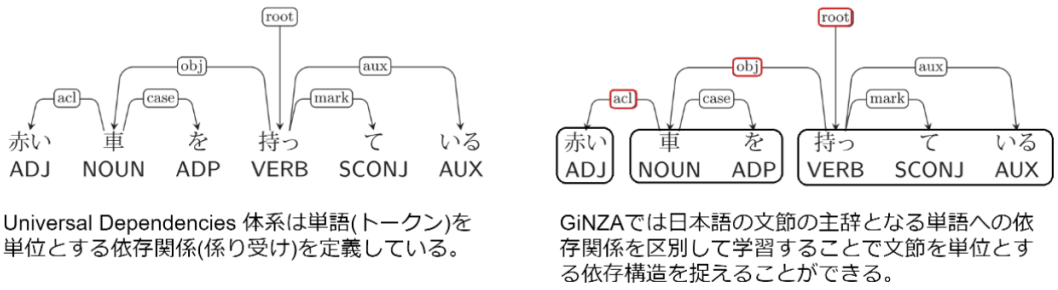


Figure 9: Word-based tokenization vs. Busetsu-based tokenization

Translation:

> Left graph: The Universal Dependencies system defines dependencies (affiliation) on the basis of words (tokens).
>
> Right graph: GINZA can distinguish and learn the dependency relationships of Japanese phrases (*bunsetsu*) to head words, thereby capturing dependency structures that treat phrases as units.

All the documentation and resources related to GiNZA can be found in its GitHub page and its blog, however most of it is only available in Japanese. Documentation about the functions and usage of the library was translated for the purpose of this dissertation.

4.2.3 bunkoOCR

BunkoOCR is a transformer model-based OCR tool that extracts Japanese text present in images and exports it in plain text, HTML, JSON, or AozoraBunko format using GPU resources. Several parameters can be adjusted to improve its performance. These adjustments allow for fine-tuning the recognition process for specific types of documents or image quality.

- Character Thinning:
    - `blank_cutoff`: Controls the threshold for ignoring thin lines during processing. Higher values (default 35) focus on darker, more confident characters.
- *Furigana* Recognition:
    - `ruby_cutoff`: (0.0 - 1.0) Sets the confidence level for identifying small characters (*furigana*) above *kanji*. Lower values (default 0.5) may capture faint *furigana*, while higher values reduce false positives.
    - `rubybase_cutoff`: (0.0 - 1.0) Similar to `ruby_cutoff`, but determines the confidence level for the main character (base character) associated with the furigana. A lower value (default 0.4) might recognize faint base characters.
- Text Layout:
    - `space_cutoff`: (0.0 - 1.0) Threshold for detecting spaces between characters. A lower value (default 0.75) improves separation but might miss spaces in English words.
    - `line_valueth`: (0.0 - 1.0) Confidence level for connecting characters into lines. Lowering the value (default 0.5) might recognize distant characters as part of a line.
- Character Detection:
    - `detect_cut_off`: (0.0 - 1.0) Threshold for identifying individual characters. Lower values (default 0.5) might recognize faint or unclear characters but increase misidentification.
- Image Preprocessing:
    - `resize`: Scales the input image for processing by the model. Useful for adjusting character size within the model's optimal range (roughly 15-256 pixels).
- Performance:
    - `sleep_wait`: Introduces a delay (in seconds) between processing images. Useful for managing processing load on the system.

By adjusting these parameters, users can optimize the OCR tool for specific use cases, such as handling faint text, complex layouts, or specific character types like *furigana*, which were present in the files to work with, and no other OCR tool could properly deal with.

The original the resources, as well as the above summarized features are only available in Japanese and were translated for the purpose of this dissertation.

4.2.4 Pandas

Pandas is an open-source data analysis and manipulation library for Python. It provides data structures and functions needed to work with structured data. For the development of this application, Pandas offered an easy way to read and navigate through tokens JSON files and vocabulary CSV files in order to access the pre-processed databases.

Additionally, a handy display functionality was used to properly show the user the list of potentially unknown vocabulary list when they choose their proficiency level, by setting the maximum number of display rows from the output to none and then printing the whole list of matching vocabulary words.

4.3 Data Preprocessing

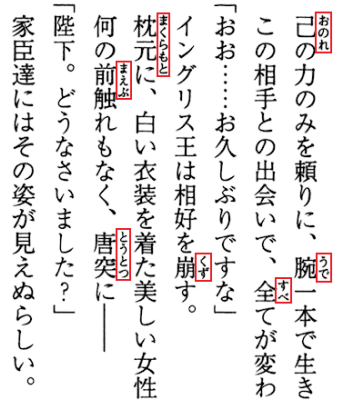
Data preprocessing was the initial part of the development of this project, done before the architecture of the application or its functions were defined. A good foundation of data was needed to generate relevant data and output from the input text given by the user. However, once the data was collected, it had to be formatted and presented in a way it could be processed properly by NLP application and libraries. Multiple challenges arose during this process, some of them being easier than others to solve.

4.3.1 Cleaning Texts

Graded reading texts from *Tadoku* were presented in PDF format, and depending on the text, they would be written horizontally from left to right or vertically from right to left. The files also contained images, page numbers, and *furigana* readings.

Different approaches were tried on how to homogenize the cleanse of the PDF files, such as multiple OCR tools, creating a script to convert PDF files into TXT files, converting the PDF files into other formats such as HTML. Ultimately, the most effective way to do so was to convert each PDF page to JPG image using a Python script, and then using an OCR tool called BunkoOCR to extract the text in the images. Despite it being a simple and straight forward task, it was not easy to find an OCR tool capable of dealing with *furigana* reading notations in an effective way. Most OCR tools tried were unable to differentiate between *kanji* and *furigana*, duplicating words when extracting the text from the images and causing a duplication of most of the words present in the text. Others, which were supposedly trained to deal with Japanese language, would often come up with text that was not present in the original files, adding hallucinations to the original texts and modifying it considerably.

It is worth mentioning that BunkoOCR was the only tool found capable of dealing with this problem. *Furigana* is a reading aid used in Japanese to display the reading of complex and uncommon *kanji*. It places *hiragana* readings above the *kanjis*:



Every OCR tool used to perform this task failed, except BunkoOCR. Based on a pretrained LLM model to perform this task, the tool managed to offset the position of the *furigana* within the actual text giving as a result a plain text without these readings. This problem was quite relevant for the development of the application, as a text with *furigana* would have extra tokens that are not needed and irrelevant, and it was not realistic to remove these *furigana* notations manually in 1796 pages.

Figure 10: Furigana text marked by red boxes

Once BunkoOCR was used, the 1769 TXT files were merged, creating 5 different TXT files each of them corresponding to one of the levels of complexity. These texts were further cleansed by removing the extra unnecessary white spaces and line breaks and adding a line break at the end of each sentence, this was done just to improve the human readability of the texts files. Additionally, RegEx scripts were used to remove all the present ASCII error characters after the OCR, and all non-Japanese characters such as alphabet letters and numbers, which would be of no use when comparing the future tokens against Japanese vocabulary lists. The used code snippet was the following:

```
for line in in_file:
    text2 = re.sub(r'[\ufffd\ufffb\ufff9\u3000\u00a9]', '', line)
    text3 = re.sub(r'[a-zA-Z0-9 ]', '', text2)
    out_file.write(text3)
```

With this process, a total of 1796 PDF pages were converted into images and the text in them was extracted into TXT files which were later merged together, prepared for human readability, and cleansed of non-Japanese, numerical, and error characters.

4.3.2 Tokenization

The tokenization process began by developing a script capable of utilizing GiNZA library to tokenize the cleansed texts and produce a JSON file containing for each of the tokens produced, the token text, the normalized form, the lemmatized form, and it's part of speech. As mentioned before, GiNZA offers much more sophisticated functions, but these were not needed, and any extra data would slow the script and unnecessary increase the size of the files.

In that regard, the script had to be adjusted because GiNZA 5.1 library makes use of SudachiPy to tokenize the text, and SudachiPy cannot process texts over 49149 bytes. According to sudachi.rs's code, the maximum text length in bytes is defined as `u16::MAX / 4 * 3`, which equals to 49149. Therefore, if any given text is bigger that this size in bytes, GiNZA would raise and input error. According to Sudachi's developers, this number was chosen based on performance. This problem was solved by adding the following snippet to the script:

```
def tokenize_text(text, max_length=49149):
    # Divide the text into segments of a maximum length. Ginza cannot
process texts over 49149 bytes
    segments = []
    segment = ''
    for word in text.split():
        if len((segment + word).encode('utf-8')) > max_length:
            segments.append(segment)
            segment = word + ' '
        else:
            segment += word + ' '
    segments.append(segment)
```

These tokens would be saved in a JSON file, and the same code would be used to later tokenize the input text given by the user. It is important to mention that this code takes into consideration token repetitions. Before solving this problem, JSON files would contain almost triple the number of tokens. This was because common words, proper names, conjunctions, punctuation and other tokens would repeat with a text thousands of times. To achieve so, the items were saved into a set, avoiding duplications. This problem was not only being inefficient in terms of file size and processing time, but also adulterated considerably the metrics.

These tokens, both the graded reading ones and the input text ones would be saved and compared against each other, giving us the percentage of matching lemmas from the input text present in the graded reading texts, and allowing us to give an estimation of the level of the input text.

Once the tokens were created, the JSON file looked like this:

And having a total of:

- 2439 tokens from the graded reading beginner level
- 2436 tokens from the graded reading starter level (JLPT 5)

- 2383 tokens from the graded reading upper-beginner level (JLPT 4)
- 1769 tokens from the graded reading lower-intermediate level (JLPT 3)
- 2442 tokens from the graded reading intermediate level (JLPT 2)

4.4 Kanji Statistics

The number of *kanjis* present in the input text was calculated with a RegEx expression that identifies every unique *kanji*. A very *kanji*-dense text can be interpreted as a high-level text for different reasons, such as a very specialized (medicine, engineering, etc.) or a text with a usage of eloquent speech (novels, science fiction).

The output consists of a print of how many of the total characters in the input text are *kanjis*, and how many unique *kanjis* are in this text.

```
def count_kanji(text):
  text_length = len(text)
  # Kanji Unicode range: \u4e00-\u9faf
  kanji_pattern = re.compile(r'[\u4e00-\u9faf]')

  # Find all kanji characters in the text
  kanji_characters = kanji_pattern.findall(text)
  kanji_characters_len = len(kanji_characters)
  kanji_characters_percentage = (kanji_characters_len / text_length) * 100
```

The same was done with *kana* characters:

```
def count_hiragana(text):
  text_length = len(text)
  # Hiragana Unicode range: \u3040-\u309f
  hiragana_pattern = re.compile(r'[\u3040-\u309f]')

  # Find all hiragana characters in the text
  hiragana_characters = hiragana_pattern.findall(text)
  hiragana_characters_len = len(hiragana_characters)
  hiragana_characters_percentage = (hiragana_characters_len / text_length) * 100
```

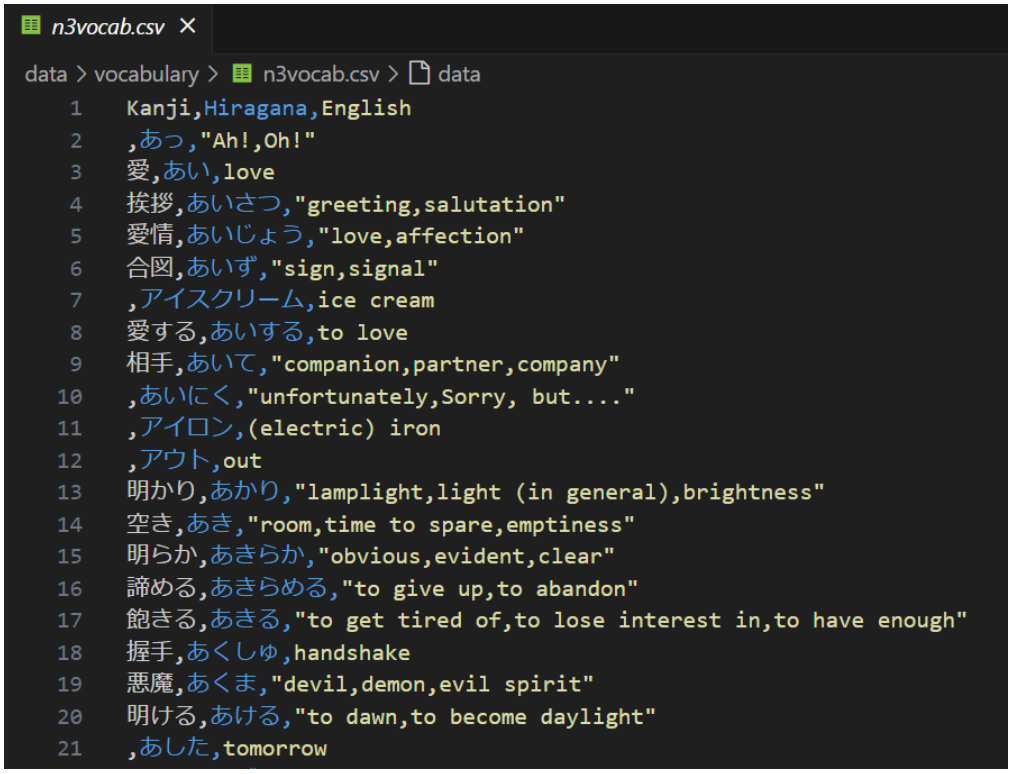
4.5 Vocabulary Extraction

Vocabulary extraction from the text can be considered the most attractive function to the user in terms of learning possibilities. Being able to extract the terms that are potentially unknown by the user offers a great way to quickly filter the text into known and unknown words.

Using the tokens generated after tokenizing the input text and comparing them against the vocabulary list that are organized into JLPT level, the user can get a detailed list of the words that are present in the text, classified into JLPT levels, with their *kanji* form, their *kana* form,

and their English equivalent. Additionally, the user received a percentage of how many of these words are in each list, a percentage of how many are *kanji* and a percentage of how many are *kana*.

These vocabulary list had to be cleaned in to contain the essential information for this application, which are *kanji* reading, *kana* reading, and English meaning, looking like this:



As it can be seen in lines 2, 7 and 10, among others, some words lack *kanji* pronunciations and only have *kana*. This was important to keep in mind with iterating through these CSV vocabulary lists to get matching percentages, as each lemma had to be compared against both the *kanji* field and the *kana* field to ensure its presence in the lists and get accurate matching percentages.

5. Application Architecture

5.1 Overview of the Application

This application is an easy-to-use and effective tool that allows Japanese learners to quickly obtain some useful metrics and data that helps them estimate how complex a given text is. The interactive interface from the terminal looks like this.

```
Welcome to the Text Processing Tool!
This tool will clean and tokenize your text files.
Please provide the input file with the text to process: ../data/news/news.txt
Input file: ../data/news/news.txt
Do you want to save the tokenized text to a JSON file? (y/n): n
Completed: 100.00%
What is your Japanese knowleged level? If you don't provide it, it will default to N5.
Please provide your Japanese level (N5, N4, N3, N2, N1): N3
Your Japanese level is: N3

You can choose one of the following options:
        1. Compare the tokenized text with a list of tokens.
        2. Compare the tokenized text with a list of vocabulary.
        3. Count the number of bunsetsu in the text.
        4. Get the number of Kanjis and Hiragana characters in the text.
        5. Exit the program.
Please choose an option:
```

The four different options are:

1. Compare the text against *Tadoku* graded readings for learners: this option compares the lemmas of the input text against the lemmas of the graded readings, giving percentages of coincidences.

```
You can choose one of the following options:
        1. Compare the tokenized text with a list of tokens.
        2. Compare the tokenized text with a list of vocabulary.
        3. Count the number of bunsetsu in the text.
        4. Get the number of Kanjis and Hiragana characters in the text.
        5. Exit the program.
Please choose an option: 1

--------------------
lvl0Tokens.json
        Total matching: 94 - 42.53%
lvl1Tokens.json
        Total matching: 19 - 8.60%
lvl2Tokens.json
        Total matching: 13 - 5.88%
lvl3Tokens.json
        Total matching: 11 - 4.98%
lvl4Tokens.json
        Total matching: 20 - 9.05%
Total matching: 157 - 71.04%
--------------------
```

2. Compare the text against JLPT vocabulary lists: this option compares the lemmas of the input text against JLPT vocabulary lists divided by level, from N5 (lower) to N1

(higher). Once the comparison is done, the user is asked if they want the list of vocabulary above their level to be printed and exported.

```
Please choose an option: 2

---------------------
n1vocab.csv
        Total matching Kanji: 15 - 6.52%
        Total matching Hiragana: 33 - 14.35%
        Total cumulative matching in this file: 48 - 20.87%
n2vocab.csv
        Total matching Kanji: 8 - 3.48%
        Total matching Hiragana: 16 - 6.96%
        Total cumulative matching in this file: 24 - 10.43%
n3vocab.csv
        Total matching Kanji: 36 - 15.65%
        Total matching Hiragana: 34 - 14.78%
        Total cumulative matching in this file: 70 - 30.43%
n4vocab.csv
        Total matching Kanji: 19 - 8.26%
        Total matching Hiragana: 17 - 7.39%
        Total cumulative matching in this file: 36 - 15.65%
n5vocab.csv
        Total matching Kanji: 39 - 16.96%
        Total matching Hiragana: 33 - 14.35%
        Total cumulative matching in this file: 72 - 31.30%
Total unique matching Kanji: 102 - 44.35%
Total unique matching Hiragana: 67 - 29.13%
---------------------
Do you want to get the vocabulary higher that the level you provided? (y/n):
```

And a sample of the vocabulary:

```
N1 vocabulary:

Kanji Hiragana                                                                                                  English
   私      あたし                                                                                                  I (fem)
  溢れる    あふれる                                                                                  to flood, to overflow, to brim over
   或る      ある                                                                                        a certain..., some...
  NaN      いく                                                                                          to come, to orgasm
  何時も     いつも                                                                  always, usually, every time, never (with neg. verb)
   未だ     いまだ                                                                                  as yet, hitherto, not yet (neg)
   入る      いる                                                  to get in, to go in, to come in, to flow into, to set, to set in
  起こす     おこす                                                                          to raise, to cause, to wake someone
   俺       おれ                                                                  I (ego) (boastful first-person pronoun)
   仮        か                                                                          tentative, provisional
   科        か                                                                             department, section
   個        か                                                                                 article counter
   嘗て      かつて                                                                                    once, ever
   街       がい                                                                                  ~street, ~quarters
 くっ付く     くっつく                                                                          to adhere to, to keep close to
   包む      くるむ to be engulfed in, to be enveloped by, to wrap up, to tuck in, to pack, to do up, to cover with, to dress i
   頃       けい                                                                  time, about, toward, approximately (time)
   児        こ                                                                              child, the young of animals
```

```
N2 vocabulary:

Kanji Hiragana                          English
   在る       ある                    to live,to be
   煎る       いる                   to parch,to fry
   炒る       いる                              NaN
    蚊        か                         mosquito
   改札     かいさつ             examination of tickets
    殻       から              shell,husk,hull,chaff
  NaN     くっつく       to adhere to,to keep close to
   請う       こう                 to ask,to request
   高層     こうそう                           upper
    琴       こと               Koto (Japanese harp)
  NaN     こぼれる               to overflow,to spill
   湿る       しめる  to be wet,to become wet,to be damp
   水滴     すいてき                     drop of water
   隙間       すきま          crevice,crack,gap,opening
   刷る       する                          to print
   通勤     つうきん                 commuting to work
 懐かしい     なつかしい               dear,desired,missed
   生る       なる                     to bear fruit
   匂う       におう    to be fragrant,to smell,to stink
   載る       のる to appear (in print),to be recorded
  NaN    マンション     large apartment,apartment house
   漸く     ようやく           gradually,finally,hardly
```

3. Count the number of *bunsetsus* (syntactic units) present in the text: this option is really useful, as it breaks the text down into sentences, and each sentence is then divided into *bunsetsus*. A *bunsetsu* is a syntactic unit that can be composed of multiple words. As explained over this dissertation, Japanese language lack of spaces makes it unclear where to divided words, and in terms of syntactic analysis, it is complicated for learners to identify where a word begins and ends.
   The output would look like the following:

```
---------------------
Sentence: 懐かしい声と匂い、愛おしい光と温度。
List of Bunsetsu: [懐かしい, 声と, 匂い、, 愛おしい, 光と, 温度。]
Bunsetsu count: 6

Sentence: 私は大切なだれかと隙間なくぴったりとくっついている。
List of Bunsetsu: [私は, 大切な, だれかと, 隙間なく, ぴったりと, くっついている。]
Bunsetsu count: 6

Sentence: 分かちがたく結びついている。
List of Bunsetsu: [分かちがたく, 結びついている。]
Bunsetsu count: 2

Sentence: 乳房に抱かれた乳ち呑のみ児の頃のように、不安や寂しさなんてかけらもない。
List of Bunsetsu: [乳房に, 抱かれた, 乳ち呑のみ, 児の, 頃の, ように、, 不安や, 寂しさなんて, かけらも, ない。]
Bunsetsu count: 10

Sentence: 失ったものは未いまだひとつもなく、とても甘やかな気持ちが、じんじんと体に満ちている。
List of Bunsetsu: [失った, ものは, 未, いまだ, ひとつも, なく、, とても, 甘やかな, 気持ちが、, じんじんと, 体に, 満ちている。]
Bunsetsu count: 12
```

   This would allow the user to properly know how to properly parse and interpret each part of the sentence, helping them identify syntactic elements such as subjects, verbs, direct objects, or indirect objects.

4. Get the number of *kanji* and *kana* characters in the input text: this is a simple function that tells the user the number of *kanji* and *kana* characters respectively in the input text.

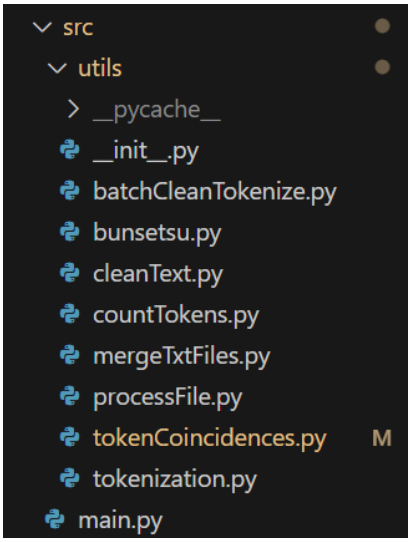
```
Please choose an option: 4

--------------------
The text length is: 896 characters and the number of hiragana characters is: 514 - 57.37% of the text
The text length is: 896 characters and the number of kanji characters is: 213 - 23.77% of the text
--------------------
```

5.2 Desing and Implementation

The application is written in Python and designed around the use of functions.

A main (**main.py**) is implemented to execute the program and an init method **(__init__.py**) was created to initialize all the functions implemented in the program.

A source folder (**src**) is implemented, which contains the main and all the functions withing the folder "**utils**". The "**utils**" folders contains Python script files for each of the functions developed and used in this application. These functions are:



BatchCleanTokenize.py is a script that combines both the **tokenization.py** and the **cleanText.py**. It is designed to perform both tasks at the same time. The script removes Unicode error characters, removes alphanumeric characters and extra white spaces, removes line feeds, inserts line feeds after the occurrence of "。", and saves it to an output directory.

Then, it is tokenized with **tokenization.py** using GiNZA's language model "**ja_ginza_electra**", and tokenizing the text passing four of GiNZA's preexisting variables as arguments, which are "**token.text_**", "**token.norm_**", "**token.pos_**", and "**token.lemma_**". The text is divided into segments smaller than 49149 bytes (previously explained), tokenized with GiNZA, the tokens are added to a set to avoid duplicates, and then converted again to a dictionary to be exported in JSON format.

MergeTxTFiles.py was used to combine the result of BunkoOCR, as each PDF page from *Tadoku* graded readings was converted into JPG, and then OCRed into TXT files. These TXT were merged together with this script.

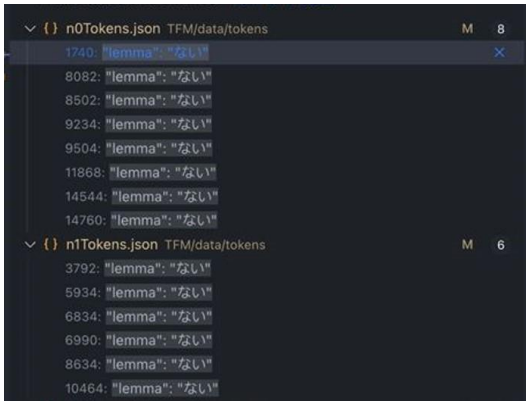
ProcessFile.py is a script that has two reading functions: one for JSON files and another one for CSV files. These functions make use of Pandas to get the data frames from the JSON and the CSV databases.

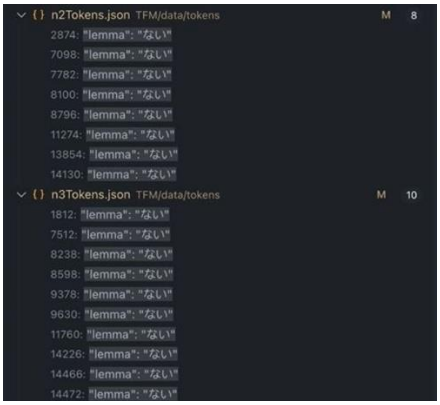
TokenCoincidences.py is one of the main scripts which is in charge of comparing the input tokenized text lemmas against the lemmas from the tokenized *Tadoku* graded readings, and against the JLPT (N5-N1) vocabulary lists. This script has three functions, which are the following:

- Function 1: '**match_tokens (input_dataframe, token_dataframes)**'

This function matches tokens (concatenations of lemma and part-of-speech tags) between the 'input_dataframe' and a dictionary of 'token_dataframes'. This function was needed to solve a problem regarding repetitions. Because of the nature of Japanese language and GiNZA tokenizer, some words would be tokenized multiple times having different POS tags, and this affected the metrics when comparing input lemmas against the database.

For example, the word "ない" which literally means "no" and it is used to negate, was considered to be up to 10 different tokens (words) in a single JSON file. The reason was the fact that it was being saved as different due to its different writing (*kanji* and *kana*), and most importantly because of its different POS. Depending on the usage and depending on the tokenizer interpretation, a word can have multiple POS, hence the multiple tokens. The agglutinative nature of Japanese also impacted heavily on this type of tokens, as suffixed are added to words, in this case the negation, and depending on the context, it can be considered different POS.





Concatenating both the lemmas and the POS and converting them into a set, allowed to iterate through the input data frame avoiding repetitions and resulting in a much more exact percentages of matching lemmas.

- Function 2: **'match_lemmas(input_dataframe, vocabulary_dataframes)'**

This function matches lemmas from the 'input_dataframe' against two specific colums in the 'vocabulary_dataframes', which are the previously explained columns from the CSV '*Kanji*' and '*Kana'.*

Comparing against both columns is key as some words ofter have *kana* representation but lack *kanji* representation. Additionally, a text that aims for simplicity might use *kana* versions of words that are often written in *kanji,* hence, the importance of iterating over both columns until a match is found.

- Function 3: **'match_vocabulary(input_dataframe, vocabulary_dataframe, language level)'**

This function matches lemmas from the 'input_dataframe' against vocabulary data of different JLPT levels (N5-N1). This function makes use of the user's JLPT level inputted in **main.py**, passing it as an argument and creating a list of matching vocabulary which

contains words present in the text that are above the user's level, displaying the matching vocabulary entries.

Bunsetsu.py script contains a function that divides the text into sentences and counts the number of bunsetsus in each sentence. As explained before, *bunsetsus* are syntactical units present in Japanese that could be defined as phrases. These phrases have meaning by themselves and are consist of a content word and zero or more function words.

The script contains a function capable of counting the amount of *bunsetsu* per sentence, printing them to the user and allowing them to break down potentially complex sentences into easier to understand chucks. GiNZA's language model "**ja_ginza_electra**" is imported with SpaCy to perform this task, and GiNZA's function "**bunsetsu_spans**" is used to parse the *bunsetsu*.

6. Evaluation

6.1 Testing Data

Two different texts were used for testing the application, a news article, and a fragment from a chapter of a novel. These two texts could be considered high-level Japanese texts aimed for native speakers and represent material that could potentially be object of interest of a Japanese learner.

The first one is a news article from the newspaper *Asahi Shimbun* titled: Asian Security Conference: dialogue and long-term stability (アジア安保会議　対話重ね長期の安定へ). The news despite being short in terms of length, it is full of specialized terms and deals with the celebration of a conference in Singapore, where the ministers of defence of most East-Asian and Pacific countries attended to deal with the situation of the growing tensions between China and the US in Taiwan and surrounding areas and aiming for peace and respectful relations between the countries in the area in order to achieve long term stability. The text has not *furigana*, which means that the reader has no way to read uncommon and rare words that they don't know prior to the reading.

A Japanese learner should be at least intermediate-advanced to grasp the general idea of what is being said in this article and have an advanced *kanji* foundation to fully understand the idea presented, and probably many terms would have to be consulted in the dictionary to make sure the text is being understood. An intermediate Japanese learner would have serious problems understanding this text, and a native Japanese speaker might have some difficulties reading certain *kanjis* depending on their literacy level and their educational background.

The second text used for this project is an extract from the novel Your Name (君の名は。) written by Makoto Shinkai (新海誠). This novel is extremely popular in Japan mainly within younger audiences and its author is considered to be one of the revelation authors of this era. The novel deals with young romance. The language used in this novel is easy to understand and very alike to everyday language, however, the author occasionally makes use of non-common *kanji* characters to express subtle nuances that differentiate from common *kanjis*. These characters can present a difficulty for readers and often times are accompanied by *furigana* readings so the reader can understand the meaning of the word even if they don't know the *kanji*.

These are the few difficulties that a Japanese learner could confront dealing with this text, and overall, an intermediate Japanese learner could read through it at a moderate pace without many difficulties and some dictionaries look ups. An advanced Japanese learner would have no problems understanding this novel with minimum dictionary look ups and for a Japanese native speaker it would present zero difficulties.

6.2 Evaluation Results

The results obtained from analysing both texts are the following:

Token coincidences:

	Level 0	Level 1	Level 2	Level 3	Level 4	Total
News article	70 tokens 25.36 %	14 tokens 5.07 %	7 tokens 2.54 %	6 tokens 2.17 %	12 tokens 4.35 %	109 tokens 39.49 %
Novel fragment	94 tokens 42.53 %	19 tokens 8.60 %	13 tokens 5.33 %	11 tokens 4.98 %	20 tokens 9.05 %	157 tokens 71.04 %

Figure 11: Token coincidences between input texts and Tadoku graded readings

Vocabulary coincidences:

	JLPT N5	JLPT N4	JLPT N3	JLPT N2	JLPT N1	Total
News article	17 kanji 5.76 %	13 kanji 4.41 %	49 kanji 16.61 %	11 kanji 3.73 %	43 kanji 14.58 %	133 kanji 45.09 %
	23 kana 7.80 %	11 kana 3.73 %	29 kana 9.83 %	22 kana 7.46 %	22 kana 7.46 %	107 kana 36.25 %
	40 total 13.56 %	24 total 8.14 %	78 total 26.44 %	33 total 11.19 %	65 total 22.03 %	240 total 81.36 %
Novel fragment	39 kanji 16.96 %	19 kanji 8.26 %	36 kanji 15.65 %	8 kanji 3.48 %	15 kanji 6.52 %	117 kanji 50.87 %
	33 kana 14.34 %	17 kana 7.39 %	34 kana 14.78 %	16 kana 6.96 %	33 kana 14.35 %	133 kana 57.82 %
	72 total 31.30 %	36 total 15.65 %	70 total 30.43 %	24 total 10.43 %	48 total 20.87 %	250 total 108.69 %

Figure 12: Coincidences between input texts and vocabulary lists by JLPT level

```
---------------------

You can choose one of the following options:
        1. Compare the tokenized text with a list of tokens.
        2. Compare the tokenized text with a list of vocabulary.
        3. Count the number of bunsetsu in the text.
        4. Get the number of Kanjis and Hiragana characters in the text.
        5. Exit the program.
Please choose an option: 4

---------------------
The text length is: 1023 characters and the number of hiragana characters is: 424 - 41.45% of the text
The text length is: 1023 characters and the number of kanji characters is: 450 - 43.99% of the text
---------------------
```

Figure 13: News article number of characters in kanji and kana

```
You can choose one of the following options:
        1. Compare the tokenized text with a list of tokens.
        2. Compare the tokenized text with a list of vocabulary.
        3. Count the number of bunsetsu in the text.
        4. Get the number of Kanjis and Hiragana characters in the text.
        5. Exit the program.
Please choose an option: 4

---------------------
The text length is: 896 characters and the number of hiragana characters is: 514 - 57.37% of the text
The text length is: 896 characters and the number of kanji characters is: 213 - 23.77% of the text
---------------------
```

Figure 14: Novel fragment number of characters in kanji and kana

7. Results and Discussion

7.1 Overview of Results

Overall, the results obtained from the two texts align with what was expected in terms of difficulty and both tokens and vocabulary matches.

In the first text, the news article, the number of matching tokens was a 39.49 %, paying special attention to a 25.36 % of tokens from the Level 0 graded readings (the lowest level). The nature of the graded readings, as explained in previous sections, is one of "artificially crafted" texts aimed to Japanese language learners. This text contain meticulously chosen words which are of high frequency, specially in the lowest levels, to help learners build a good foundation of the language. This news article is a complex text with highly specialised vocabulary which is not present in these graded readings, hence, the less than 40 % matching coincidence with most of them being high frequency words.

In terms of vocabulary, an 81.36 % of the words are present in the vocabulary data set, from which 45.09 % are *kanji* words and 36.25 % are *kana* words. This gives a great insight on how highly specialised and formal text like a news article tend to use a high number of *kanji* words. Furthermore, a 26.44 % of the words are from the N3 level (intermediate), the summatory of the N3, N2, and N1 words make up 59.66 % of the words in the text, and only a 13.56 % of the words are from the N5 level (beginner). This data furtherly clarifies how the nature of the text in not one that would be suitable for a beginner, but rather a text for a intermediate-high student that would be around N2 level.

Lastly, when comparing the ratio of *kanji* and *kana* number of characters, the results obtained indicate that a 43.99 % of the characters were *kanji* while a 41.45 % of the characters were *kana.* These results indicate once again that the number of *kanji* characters is high enough for this text to be considered high-difficulty level and for a learner to have a solid language foundation and a high level of *kanji* knowledge to be able to understand the idea behind the text, and a deep understanding of Japanese language to be able to fully grasp the overall meaning of this article.

On the other hand, the second text results are quite different. The number of matching tokens was a 71.04 %, which is surprising due to the previously mentioned fact of these grading texts being carefully crafted to contain only common words, characters, and structures. This first result already gives a hint of the overall complexity of the text not being extremely high.

Vocabulary wise, a 108.69 % of the words present in the text are part of the vocabulary lists. The reason for this result being over a 100 % will be discussed in the next section as is a problem that has been identified and partially fixed but not yet completely. However, this high number of coincidences is still a valuable result and shows that all words present in the text were found in the vocabulary list, hence no complex or extremely specialised words from outside the lists were present in the fragment extracted from this novel. As explained previously, this is a novel aimed at teenagers and is written in a very casual style, therefore its low level of complexity. In this novel fragment, a 57.82 % of the words were written in *kana*, and a 50.87 % in *kanji*, this ratio shows that *kanji* words are not used as much as the news article, but still the ratio is close to fifty-fifty. Additionally, a 31.30 % of the words present in the text correspond to the N5 level (beginner), and a 77.38 % of the words form a solid base

from the N5, N4, and N3 levels, with only a 31.30 % being part of the N2 and N1 levels (advanced).

Finally, in terms of *kanji* and *kana* characters, this novel fragment had a 57.37 % of individual *kana* characters and only a 23.77 % individual of *kanji* characters. These results provide a clear indication that the level of complexity of this text is not as high as the news article. A Japanese language student with a solid foundation on the language and a wide knowledge of *kanji* characters would have close to zero problems reading this text (N2 and N1), it would be an appropriate reading for an intermediate student (N3), and it would be challenging for lower proficiency students and beginners (N4 and N5).

7.2 Discussion of Findings

Overall, the results are very straightforward and provide clear insights of the complexity of the input texts. Simply by taking a close look at the *kanji-kana* word ratio, the *kanji-kana* character ratio, and the distribution of words through the different JLPT vocabulary lists a good estimation of the level can be guessed. However, and as observed in the novel fragment, the percentages of matching vocabulary words are not exactly accurate and are higher than they should be.

This problem was identified early in the development of the application, as when comparing the tokenized input text tokens lemmas against the vocabulary list entries, homophones (same pronunciation and same *kana* writing) would be flagged multiple times, give up to 20 and more coincidences in some cases and resulting in 250 % matching words in some tested texts. Additionally, repeated words, such as proper names, were being counted multiple times. This was solved by saving the coincidences in a set, avoiding repetitions, and taking into consideration the POS of the lemmas as another comparing factor to avoid repetitions as much as possible.

This, however, was not enough to solve the problem. When it comes to vocabulary lists, lemmas would be iterated though both *kanji* and *kana* columns of all the 5 lists of vocabulary (N5 to N1), first the *kanji* and stopping of a coincidence is found, and then the *kana.* However, it was soon realised that multiple homophones were being triggered across all the vocabulary lists. When a word is written in *kana* and the lemma iterates though the *kana* column, it stopped on the first result, and that one was not always the needed one. As a result, it was decided to include in the results all the matching homophones for a *kana* word, resulting in a higher % of results. This would happen with all the five lists, so for example, the word そう *sou* is present multiple times in the results because of it being a basic and highly used affix word with multiple meanings, being present in all the vocabulary lists with multiple meanings and usages.

This problem was solved in the token lists by taking into consideration the POS of the lemmas. However, the vocabulary lists are in CSV format and only contain three fields: *kanji, kana,* and English equivalent. For this reason, a dictionary in JSON format was considered to be implemented somehow as reference to further compare the input tokenized text lemmas and have more context, but this idea was soon dismissed at it required a huge amount of work which is out of the scope of this project at the moment. Additionally, it would be of high interest to use some kind of NLP tool or library capable of identify patterns and contextualise words through the understanding of the previous and following tokens, assessing the possibilities and only choosing the homophone with the highest confidence interval.

7.3 Limitations and Challenges

During the development of this project multiple limitations were found and a lot of challenges had to be overcome. Some of them could not be solved and it is planned to implement solutions in future versions of the application. Some examples were mentioned throughout this dissertation such as the need to combine CSV Vocabulary list with Dictionaries to increase the amount of information given to the user, such as Part-of-speech. Other intended improvement of the application is to implement a graphic interface where the user can simply paste the text that wants to analyse and have different buttons that allow them to use the different functions present in the application. Finally, an automatic translation could be implemented to the application. A simple API could be implemented, but also a corpus could be downloaded, then aligned, and with this a small automatic translation that works for Japanese-English language combination could be trained.

Regarding problems that were solved, all the relevant ones were already described. First one was extracting text from PDF files in an efficient way. Dealing with *furigana* readings was a real drawback and at some point, it seemed impossible to overcome, but thanks to proper research (in Japanese language) BunkoOCR was found and implemented, solving the problem in the blink of an eye.

Another problem was processing extremely long texts with the GiNZA tokenizer, as the Python has limitations in terms of size, but this was also solved by splitting the text in smaller chunks before processing it, however, thanks to a comment in GitHub by another user it could be solved.

The third main challenge was to notice that tokens were being repeated thousands of times within the same texts. It is something that makes sense, but at that time it was not easy to realize that elements such as proper names, common words, connectors, conjunctions, or punctuation symbols were being repeated over thousands of times. That meant JSON files with over 70.000 tokens. This was fixed by simply modifying the code, so it skips tokens that have already been added to the list of tokens. It was a real improvement not only in speed but also in accuracy. Before fixing this issue, the metrics were not accurate as each of these repeated tokens would trigger positive when comparing, resulting in adulterated results with higher matching percentages, at some point close to 95% and 99%.

Finally, and as explained before, this project still needs some tool or NLP technology capable of contextualising lemmas in a way homophones are properly addressed giving the user the exact word they are looking for.

8. Conclusion

After this project, the conclusions taken from this dissertation was indeed reassuming the existing problems that Japanese Natural Language Processing faces when compared with most languages.

First of all, the nature of Japanese language makes it extremely difficult to begin to work with a Japanese text. The presence of three different writing systems, the lack of white spaces that work as natural delimiters, the agglutinative aspects of the language, or the multiple ways of writing the same word. All these combined together meant that multiple solutions had to be developed to solve possible problems that could arise.

In second place, preprocessing all the data needed to create a size-acceptable and high-quality database was surprisingly unexpected. Despite the high quality of the raw data collected, numerous problems arose during the cleaning and preprocessing of these files. File formats, dictionaries structures, optical recognition of characters, or repetitions in datasets where the problems described in previous sections. These problems were solved to a certain degree as they have a direct impact on the quality of the metrics and information given to the user, but to achieve a complete and efficient fix a better approach and understanding of programming would be needed.

Finally, and probably the biggest barrier to understand the complex topic presented in this project, is the lack of research and bibliography and resources in any language that is not Japanese. Most of the resources used in this dissertation were taken directly from Japanese forums, GitHub repositories, or other websites that specialize in NLP. Some of them were poorly translated into English, but most of them were only in Japanese. Furthermore, the information about Japanese NLP in English was very limited and outdated, and it was almost non-existent in Spanish. This, however, makes perfect sense with what was stated in this dissertation: the majority of the problems presented during the development of this application are not present in NLP for other languages (with very few exceptions). English and other European languages NLP researchers do not have the necessity to investigate about these problems, hence, there is no bibliography about it but only the one written by Japanese NLP researchers.

9. Bibliography

- Asahi Shimbun. (2024, June 6). Asian Security Conference: dialogue and long-term stability (社説）アジア安保会議　対話重ね長期の安定へ). Only available in Japanese. https://www.asahi.com/articles/DA3S15950229.html?iref=comtop_Opinion_04
- EDRDG. Electronic Dictionary Research and Development Group. Last accessed March 11, 2024. https://www.edrdg.org/wiki/index.php/Main_Page
- Feng, Z. (2023). Formal analysis for natural language processing: A handbook / Zhiwei Feng. (1st ed. 2023.). *Springer Nature Singapore*. https://doi.org/10.1007/978-981-16-5172-4
- GiNZA – Japanese NLP Library. *GitHub*. Only available in Japanese. https://megagonlabs.github.io/ginza/
- Halpern, J. (2000). The Challenges of Intelligent Japanese Searching. The CJK Dictionary Institute, Inc. https://www.cjki.org/joa/joapaper.htm
- Indurkhya, N., & Damerau, F. J. (2010). Handbook of natural language processing, second edition.
- Japanese-Language Proficiency Test. N1-N5: Summary of Linguistic Competence. *The Japan Foundation / Japan Educational Exchanges and Services.* https://www.jlpt.jp/e/about/levelsummary.html
- Lithium0003. findtextCenterNet. *GitHub*. https://github.com/lithium0003/findtextCenterNet/tree/main
- Lithium0003. (2023). bunkoOCR (Windows). Retrieved from https://lithium03.info/product/bunkoOCR.html Only available in Japanese.
- Lithium0003. (2023). 日本語 OCR への道 Nihongo OCR he no michi [Towards the path of Japanese OCR]. Only available in Japanese. Retrieved from https://lithium03.info/jpOCR/index.html
- Megagonlabs. (2023, September 23). GiNZA. A Japanese NLP Library using spaCy as framework based on Universal Dependencies. *GitHub*. Only available in Japanese. https://github.com/megagonlabs/ginza
- Megagonlabs. Explanation of Bunsetsu API (Bunsetsu API no Kaisetsu). GitHub. Only
- Niwashi. (2021, September 21). Introduction to Japanese spaCy/GiNZA [日本語/Eng]. *Kaggle*. https://www.kaggle.com/code/marutama/introduction-to-japanese-spaCy-ginza-eng/notebook
- NPO Tadoku Supporters. Last accessed March 11, 2024. https://tadoku.org/japanese/en/graded-readers-en/
- O'Leary, P. (2020, August 13). How to Tokenize Japanese in Python. *Dampfkraft*. https://www.dampfkraft.com/nlp/how-to-tokenize-japanese.html
- Pandas. User Guide. https://pandas.pydata.org/docs/user_guide/index.html
- Scharf, A. (2020). Japanese NLP with SudachiPy, spaCy, and GiNZA. *GLOBIS Advent*. https://qiita.com/acscharf/items/66017434ce1fc40deeb8
- Shinkai, M. (2016). Your Name (君の名は。). *Kadokawa Bunko* (角川文庫).
- Shirakawa, T. Megagonlabs. GiNZA 5.1 cannot process long texts. *GitHub*. https://github.com/megagonlabs/ginza/issues/242
- Shpika, D. (2024). JMDict-simplified. *GitHub*. https://github.com/scriptin/jmdict-simplified
- spaCy Japanese. Available trained pipelines for Japanese. *spaCy*. https://spaCy.io/models/ja
- Sugimoto, K. (2024). Awesome-japanese-llm. *GitHub*. Only available in Japanese. https://github.com/llm-jp/awesome-japanese-llm
- Tanakitrungruang, W. (2020, September 6). How Japanese Tokenizers Work. *Medium*. https://towardsdatascience.com/how-japanese-tokenizers-work-87ab6b256984

- Uchida, T. (2023, July 1). Janome. Japanese morphological analysis engine written in pure Python. *GitHub*. https://github.com/mocobeta/janome
- Waller, J. (2010). Japanese Language Proficiency Test Resources. https://www.tanos.co.uk/jlpt/

10. Appendices

News article used:

Asian Security Conference: dialogue and long-term stability (Automatic Translation)

The Korean Peninsula, the Taiwan Strait and the waters surrounding the Philippines. Confidence-building through dialogue and exchange is essential in the Asia-Pacific region because of the numerous crises that lie beneath the surface. Further efforts must also be made to link this to the realisation of a stable order in the long term.

The Asian Security Conference was held in Singapore, where defence ministers and experts from the region gathered to exchange views.

In conjunction with the conference, the defence ministers of the US and China met face-to-face for the first time in 18 months. The two countries' growing military tensions in Taiwan and the South China Sea led to heated exchanges, but it was a step forward that they agreed to set up a working group to discuss mutual contacts in times of crisis.

In the US, in particular, the ruling and opposition parties are increasingly competing in their hard-line stance towards China in the run-up to the elections. We hope that the military authorities on both sides will not only prevent unforeseen situations, but also take a calm response, keeping their distance from such political developments.

However, this should be a minimum goal. What is essentially required of the US and China, which have a heavy responsibility for the world order as well as Asia, is to improve the security environment over the medium to long term.

Of concern is the rapid build-up of nuclear forces and missile development in the region. Negotiations between the US and China over nuclear weapons have not been reported since November last year. At a time when Russia is making irresponsible nuclear threats, the US and China should work together to reduce the nuclear threat.

In his speech at the conference, Chinese Defence Minister Dong Jun criticised the tensions in the Taiwan Strait and South China Sea, saying that there is collusion and support by external forces. Although he avoided naming names, he probably wanted to say that US involvement was a problem.

However, it is an indisputable fact that the Chinese military has strengthened its presence in the South China Sea, including by turning reefs into military bases, and that this has caused tensions in the region. China must take seriously the fact that the participants in the conference pointed out the inconsistency of their words and deeds one after another.

The US, on the other hand, is also keen to build a deterrence network against China with its allies and friends, including Japan. Japan is also increasing its involvement in the South China Sea through its support for the Philippines.

The situation has led Indonesian President-elect Prabowo Indonesia to complain that the Global South (emerging and developing countries) is disillusioned by the rising geopolitical tensions.

We should listen sincerely to the voices of countries that wish to realise stability through diplomacy rather than competing for military power.

The agreement between Japan and South Korea on measures to prevent recurrence and the resumption of defence exchanges over the radar irradiation issue is a welcome move. A calm response that does not politicise the issue again and does not turn Japan-South Korea relations into a new source of instability in Asia is essential.

（社説）アジア安保会議　対話重ね長期の安定へ **(Original)**

朝鮮半島、台湾海峡、フィリピン周辺海域。アジア太平洋地域には数多くの危機が伏在するからこそ、対話と交流を通じた信頼醸成は欠かせない。これを長期的に安定した秩序の実現へつなげていくさらなる努力も必要だ。

地域の防衛担当閣僚や専門家がシンガポールに集まって意見を交わす「アジア安全保障会議」が開かれた。

会議に合わせて米中の国防相が、１年半ぶりに対面で会談した。台湾や南シナ海情勢で軍事緊張が高まる両国だけに激しい応酬になったが、危機時の相互連絡を話し合う作業部会を設けることで合意したのは一歩前進だ。

特に米国では選挙をにらんで与野党が対中強硬姿勢を競う構図が強まる。不測の事態を防ぐだけでなく、こうした政治の動向からも距離を置いた冷静な対応を、双方の軍事当局には望みたい。

もっとも、これは最低限の目標と言うべきだろう。アジアのみならず世界の秩序に重い責任を持つ米中両大国に本来求められるのは、中長期的な安保環境の改善だ。

懸念されるのは、核戦力の増強とミサイル開発が地域で急速に進んでいることだ。米中の核兵器をめぐる交渉は昨年１１月以降、伝えられていない。ロシアが無責任な核の脅しを仕掛けている折、米中は核の脅威の低減に向けて協働してほしい。

中国の董軍国防相は会議での演説で、台湾海峡や南シナ海の緊張について「外部勢力による共謀や支援がある」と批判した。名指しは避けたものの米国の関与が問題だと言いたかったのだろう。

だが中国軍が南シナ海で岩礁を軍事拠点化するなど海空で存在感を強め、域内を緊張させているのは動かしがたい事実だ。会議出席者から言行不一致を指摘する声が相次いだことを、中国は重く受け止めねばならない。

片や米国も、日本を含む同盟・友好国との対中抑止網の構築に余念がない。日本もフィリピン支援を通じて南シナ海への関与を強めている。

この状況にプラボウォ・インドネシア次期大統領が「地政学的な緊張の高まりにグローバルサウス（新興国、途上国）は幻滅している」と苦言を呈した。軍事力を競い合

うのではなく外交を通じた安定実現を望む国々の声に、真摯（しんし）に耳を傾けるべきだ。

レーダー照射問題をめぐって日韓が再発防止策と防衛交流再開で合意したのは歓迎すべき動きだ。問題を再び政治化せず、日韓関係をアジアの新たな不安定要因にしない冷静な対応が欠かせない。

Novel used:

Your Name by Makoto Shinkai (Automatic Translation)

Published by Kadokawa Bunko

Nostalgic voices and smells, lovely light and temperature.

I am perfectly attached to someone I care about, with no gaps.

I am inseparably bound to them.

Just like when I was a baby in the arms of my breast, there is no trace of anxiety or loneliness.

I have not lost a single thing yet, and a very sweet feeling fills my body.

Suddenly, my eyes open.

The ceiling.

Room, morning.

Alone.

Tokyo.

-I see.

I was dreaming.

I raise myself from the bed.

In those mere two seconds, the warm sense of togetherness that had enveloped me a moment ago is gone.

There is no trace, no aftertaste.

So abruptly, almost before I can think of anything else, tears begin to fall.

When I wake up in the morning, for some reason I am crying.

This happens to me from time to time.

And I can't always remember the dream I should have had.

I stare at my right hand, which has been wiped clean of tears.

A small drop of water on my index finger.

The dreams I had just a few moments ago, the tears that moistened the corners of my eyes for a moment, have already dried up.

Something very precious was once in my hand.

In my hand.

I don't know.

I give up, get off the bed, leave the room and head for the bathroom.

While washing my face, I stare in the mirror, feeling as if I had once been surprised by the lukewarmness and taste of this water.

A somewhat disgruntled face is looking back at me.

I tie my hair up as I stare in the mirror.

I slip my sleeves into my spring suit.

I finally put on the tie I've grown accustomed to tying and put on the suit.

I open the door to my flat and I close it.

In front of me, the landscape of Tokyo, finally familiar to me, spreads out before me.

Just as I once naturally learnt the names of the peaks of the mountains, I can now name some of the skyscrapers.

I pass through the crowded station gates, down the escalator, onto the commuter train, I board.

I lean against the door and watch the scenery flow by.

The windows of the buildings, the cars, the footbridges, the streets are full of people.

The sky is white with a hazy, flowery overcast.

Cars with a hundred people in them, trains carrying a thousand people, and the city through which those thousand trains flow.

I find myself, as usual, looking at that city, looking for someone, and only one person.

Me,

君の名は。新海誠 (角川文庫) (Original)

懐かしい声と匂い、愛おしい光と温度。

私は大切なだれかと隙間なくぴったりとくっついている。

分かちがたく結びついている。

乳房に抱かれた乳ち呑のみ児の頃のように、不安や寂しさなんてかけらもない。

失ったものは未いまだひとつもなく、とても甘やかな気持ちが、じんじんと体に満ちている。

ふと、目が開く。

天井。

部屋、朝。

ひとり。

東京。

――そうか。

夢を見ていたんだ。

私はベッドから身を起こす。

そのたった二秒ほどの間に、さっきまで私を包んでいたあたたかな一体感は消え失せている。

跡形もなく、余韻もなく。

そのあまりの唐突さに、ほとんどなにを思う間もなく、涙がこぼれる。

朝、目が覚めるとなぜか泣いている。

こういうことが私には、時々ある。

そして、見ていたはずの夢は、いつも思い出せない。

俺は涙をぬぐった右手を、じっと見る。

人差し指にのった小さな水滴。

ついさっきまでの夢も、目尻を一瞬湿らせた涙も、もう乾いている。

とても大切なものが、かつて。

この手に。

――分からない。

俺はあきらめてベッドから降り、部屋を出て洗面所に向かう。

顔を洗いながら、この水のぬるさと味にかつて驚いたことがあったような気がして、じっと鏡を見る。

どこか不満げな顔が、俺を見返している。

私は鏡を見つめながら髪を結う。

春物のスーツに袖を通す。

俺はようやく結び慣れてきたネクタイを締め、スーツを着る。

私はアパートのドアを開け、俺はマンションのドアを閉める。

目の前には、ようやく見慣れてきた、東京の風景が私の前に広がっている。

かつて山々の峰の名を自然に覚えたように、今ではいくつかの高層ビルの名前を私は言えるようになっている。

俺は混み合った駅の改札を抜け、エスカレーターを降り、通勤電車に、私は乗る。

ドアに寄りかかり、流れていく風景を眺める。

ビルの窓にも、車にも、歩道橋にも、街には人が溢れている。

ぼんやりとした花曇りの白い空。

百人が乗った車輛、千人を運ぶ列車、その千本が流れる街。

気づけばいつものように、その街を眺めながら私は、だれかひとりを、ひとりだけを、探している。

俺は、