Detecting typographic, particle and sentence formation errors in JLPT N5 sentences using a rules-based system

Ana Verulidze
Earlham College
Richmond, Indiana
averul19@earlham.edu

1 Abstract

NLP provides excellent computer-assisted language learning resources. However, it still remains somewhat underdeveloped for many languages, including Japanese. In order to improve the student learning experience, this paper aims to explore the effectiveness of a rules-based system for detecting typographic, particle, and sentence formation errors in JLPT N5 sentences.

2 Introduction

Computer Assisted Language Learning (CALL) can be referred to as the study of tools in computer language teaching [4]. Most of the CALL applications today offer learning resources, but only a small portion of them provide evaluation. This is because the evaluation process still depends heavily on the instructor and cannot be fully performed by a computer [3]. This paper proposes an improvement to Japanese CALL systems by implementing an error detection tool. This tool will focus specifically on the following:

- Typographic errors
  These types of errors refer to spelling mistakes, but only in cases when the misspelled word does not have a meaning of its own. As an example, if the word "びょういん" (Hospital) is misspelled as "ひょういん", the proposed tool will detect it as an error, but "びよういん" (Beauty salon) will be considered correct. The main challenges in this area occur due to the absence of spacing between words and the usage of Kanji characters, which combine multiple Hiragana together. For example, the above mentioned word, "びょういん" can also be written as "病院" using only Kanji.
- Particle usage
  Particles in Japanese text mark words that appear before them. Each of the particles has a set of rules explaining when and how they should be used. For example, the particle "de" follows the location of an event. The subject performing the event must be followed by either "wa" or "ga" particles, and the verb indicating the action being performed cannot be "imasu/arimasu" (to be). This part of the tool will aim to evaluate how closely such rules are followed.
- Sentence formation
  This refers to ensuring that each sentence is completely formed and ends with a predicate. for example, the sentence "私はこの喫茶店に友達と会いにきました。" (I came to this cafe to meet with a friend") is a correctly formed sentence since it ends with the verb "to come". Word omissions will also be considered in this section. For example, we can ommit "私は" (I + particle) from the above mentioned sentence without changing the meaning conveyed.

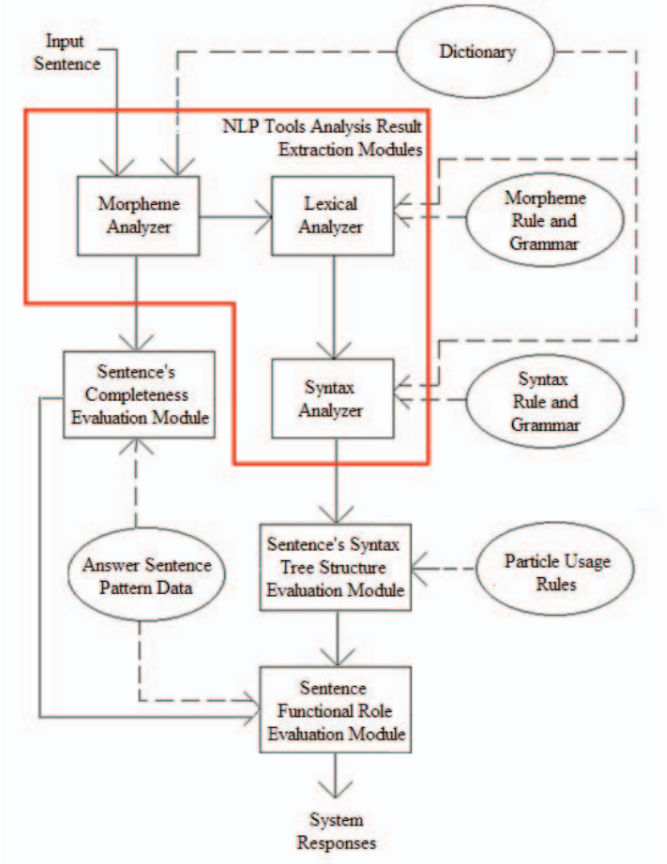

Figure 1: Software architecture of Kasmaji et al's error-detection and handling tool [3]

The error detection will be achieved through using a rules-based NLP system. A rules-based system implies the usage of linguistic rules and forms rather than statistical analysis. A similar implementation was proposed by Kasmaji et al in 2015. Their design mainly consists of two parts - implementation of the NLP tools and processing the result. This can be seen in more detail in Figure 1, where the outlined and the external sections refer to parts 1 and 2 respectively. [3].

3 Background

3.1 A brief introduction to Japanese sentence structure

Japanese sentence structure is often described as more flexible than that of many other languages. This is because the word order in which we can convey a message in Japanese can be easily modified depending on which part of the sentence we are trying to highlight. However, there are still a number of constraints we need to consider when forming a sentence, including ending it with a verb and keeping parts of the sentence within the boundaries they are held in [1]. As an example, both of these sentences are valid and translate to "I go to school by bike every day":

- "私は (I + particle) 毎ｏ (every day) 自転車で (bike + particle) 学校に (school + particle) 行きます (to go)。"
- "私は (I + particle) 学校に (school + particle) 毎ｏ (every day) 自転車で (bike + particle) 行きます (to go)。"

However, the following are not considered to be grammatically correct:

- "行きます (to go) 学校に (school + particle) 毎ｏ (every day) 自転車で (bike + particle) 私は (I + particle)。"
- "自転車で (bike + particle)行きます (to go) 毎ｏ (every day) 私は (I + particle) 学校に (to school)。"

Particle usage is one of the major parts of the Japanese sentence structure. Not only do they occur most frequently compared to other parts of speech in Japanese sentences, but they also mark complementary phrases and arguments, modify verbs and nouns, and perform semantic roles. In JLPT N5, the most common particles include the verb-modifying particles, complementizer "to", noun modifying particle "no", ga-Adjuncts, direction indicating particles "ni", and "e" [6]. Japanese spoken and written language also often uses word omissions, which refer to zero anaphora. While it is not a requirement, omitting parts of the sentence that can be recovered from context is a common practice that makes them sound more natural [3]. As an example, the sentence "私はÆを飲みます" (I am drinking water) can be shortened as "Æを飲みます" (Drinking water), as the subject of this sentence is implied. The proposed error-detection tool will attempt to detect such cases and suggest an omission.

3.2 Japanese NLP

Tokenization is an important first step in processing Japanese text for mood analytics, semantic relatedness, error detection, and more. Tokenization splits meaningful parts of sentences (that are often individual words) to prepare the text for further analysis. This process usually faces challenges that depend on the type of language used in the text. The main challenges in tokenizing Japanese texts occur because this language belongs to the unsegmented and agglutinative categories, which suggests that words have no clear boundaries and can be divided into smaller sub-parts [2]. Preprocessing is followed by morpheme, lexical and syntactical analysis of the resulting text. Kasmaji et al's error detection and handling tool uses JUMAN and KNP due to their ability to provide detailed semantic and word category-related information.

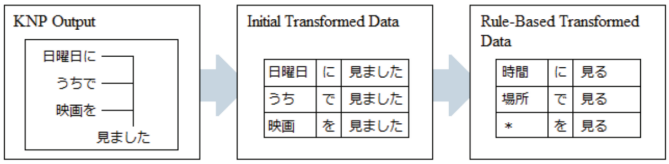

Figure 2: Steps in the Structure evaluation unit [3]

3.2.1 Result Extraction Module As a start, the Result Extraction module is implemented. This is where the NLP tools are defined and implemented. This module also ensures that the output text is modified for compatibility with the following modules. The input text is sent to JUMAN, which provides morphological analysis, and its output is then used by KNP, which generates a syntax tree represented as a table. This tree structure is depicted on the leftmost rectangle of Figure 3. It places the verb as the root, and other parts of speech as its children. In this case "ｏ曜ｏに" (on Sunday) "うちで" (at home), and "映画を" (movie) are the children of "ú ました" (to watch). Based on this information we can deduce that the input sentence was "ｏ曜ｏにうちで映画をみました。" (I watched a movie at home on Sunday). The raw analysis results are stored and modified into a structured form. The initial transformed data splits the particles "に", "で" and "を" from "ｏ曜ｏに","うちで" and "映画を" in that order and associates each word followed by a particle with the verb (watched on Sunday, watched at home, watched a movie). The Rule-Based transformation applies grammatical rules and indicates what types of words are required with each particle to form a valid sentence (time + particle "に", place + "で", * + "を". The * sign in this case indicates that there can be many options in this position). Kasmaji et al describe the smallest data structure used as a token, which stores information such as the dictionary form, part-of-speech, or inflection of meaningful morphological units. A set of tokens associated with the same sentence/phrase are referred to as a chunk, which stores data regarding its connection with a different chunk.

3.2.2 Evaluation Module Creating the syntax tree is followed by the evaluation process, which deals with sentence completeness and structure. Sentence completeness evaluation provides a brief summary of errors detected in text while later modules focus on expanding this information. The structure evaluation extracts information regarding the correctness of grammar. It covers aspects including particle usage, affix usage, verb and adjective inflections, and others. The results of structure evaluation heavily depend on the tree structure provided by the KNP module during the syntax analysis process. JUMAN and KNP allow grammatically incorrect input texts in order to produce the syntax tree, which implies the requirement of writing rules in determining the validity of the tree structure. Figure 2 gives a visual representation of this evaluation process starting with the KNP output, transforming the data, and applying grammar rules [3].

4 Design

The implementation of the error-detecting tool will have preprocessing, analytical, and evaluative phases. The preprocessing phase

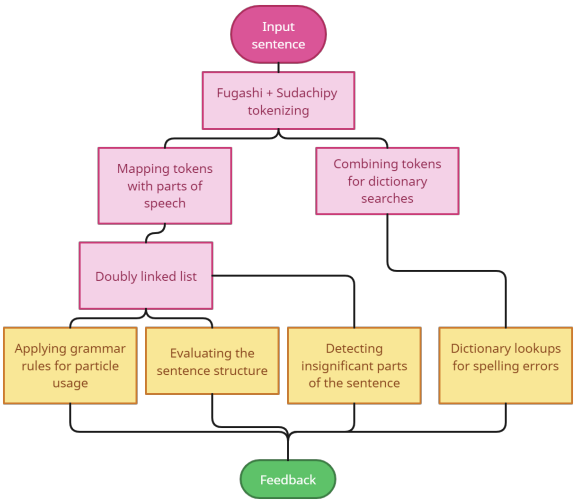

Figure 3: Data Architecture Diagram

(1) Round shape refers to data
(2) Rectangular shape refers to the process
(3) Colors from dark pink to green indicate how far we in the process we are

will cover NLP tool implementations including Fugashi [5] and Sudachipy. The analytical phase will be divided into two sections that deal with spelling errors/word omissions and sentence formation/particle usage separately. The evaluative phase will take care of describing errors and possibly proposing improvements.

4.1 Preprocessing Phase

The preprocessing phase will cover two distinct sections: a doubly linked list production phase and a mapping phase. This is where the NLP tools are defined and implemented, this module also ensures that the output text is modified for compatibility with the following modules. In the doubly linked list production phase, the input text will be tokenized using Fugashi and Sudachipy. In the mapping section, first, Fugashi and Pykakasi will be used to map each token with a part of speech. Then, some tokens may be combined to provide meaningful morphological units. As an example, Japanese tokenizing tools split “¿　館で勉Ｂしています” (I am studying at the library) as “¿　館 (library) で (particle indicating location) 勉Ｂ (study, noun) し (the root of "to do") て (te-form indicating the tense) い (morpheme) ます (polite form)”. In order to make dictionary searches simpler, “勉Ｂしています” will be combined into a single token. This will be done through the previously implemented word-splitting tool. A visual representation of the preprocessing phase can be seen in Figure 4.

4.2 Analytical Phase

In the analytical phase, the input stored as a doubly linked list structure will be processed. Fugashi will be used to determine parts of speech of each of these words and grammar rules will be applied to check accuracy. In this case, grammatically correct parts of speech

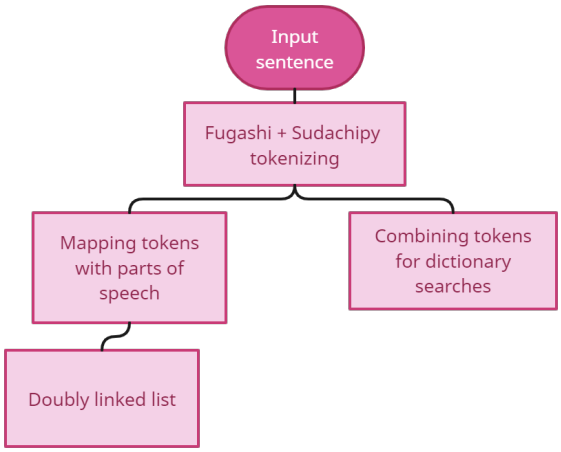

Figure 4: Steps in the Preprocessing Phase

will be compared to the extracted parts of speech. This will help evaluate how well the student uses particles. At the same time, by allowing child nodes in the list to be ordered in any way as long as they are followed by the main verb, the sentence structure validity will be determined. It is possible that sentences may contain more than one verb. However, a combined use of Fugashi and Pykakasi allows us to determine primary and secondary verbs. For example, the following sentence: “~の名前をポキにします、この名前はかわいいと思いますから” (I will name my cat Poki, because I think this is a cute name) has an auxiliary verb “します (to do, in this case - to name) and primary verb 思います (to think)”. In this phase, the combined tokens provided by the word-splitting tool will be compared to a dataset extracted from Sudachipy Dictionary, which Fugashi can directly access after importing the specific package. This implementation will use regular expressions to detect possible correct versions of the misspelled word. Using parts of speech provided by Fugashi, the subject, and object(s) of the sentence will also be determined, as well as other repetitive words. This data will be saved for further use in the feedback text. A visual representation of the Analytical Phase can be viewed in Figure 5.

4.3 Evaluation and User interface

The evaluation phase consists of combining results from the syntax tree processing and the word-splitting output processing. In this case, combining refers to extracting data from both processes and modifying it to make it user-friendly. Possible errors and solutions will be summarized together in a string that will be outputted for the user. The user interface is created using Flask since the NLP tools used are supported in Python. This web application will consist of input and output text boxes and a few additional buttons that let the user check the validity of their input text. The output text box will contain the feedback processed by the system.

5 Results and future work

In this study, an error detection tool for the JLPT N5 proficiency level of Japanese was created. web scraping was utilized to collect

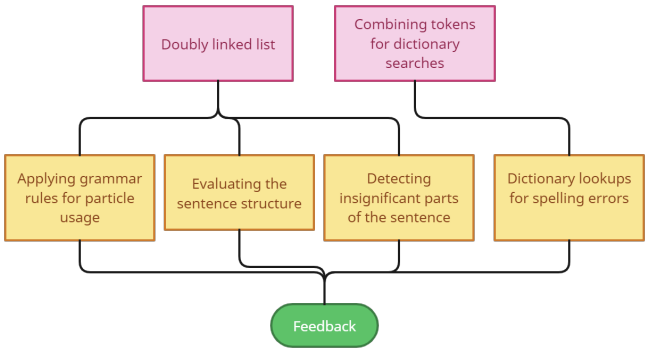

Figure 5: Steps in the analytical phase

relevant data for JLPT N5 vocabulary and grammar. Additionally, a .csv corpus of Japanese sentences was used for sentence structure and particle error evaluation. The web application allows users to hover over the text and see the parts of speech for each significant sentence component as well as receive feedback. However, it was found that the corrected version of the sentence still needs improvement since corpus lookups proved to be unsuccessful for this task.

In the future, a machine learning model will be trained to detect spelling mistakes and correct input. The tool will also be adapted to fit higher proficiency levels beyond JLPT N5.

6 Acknowledgement

I would like to express my gratitude for the unwavering guidance and constructive criticism provided by Dr. Charlie Peck, Dr. David Barbella, and Dr. Haruka Ogawa.

References

[1] Francis Bond and Timothy Baldwin. 2016. Introduction to Japanese computational linguistics. *Readings in Japanese Natural Language Processing. CSLI Publications, Stanford* (2016), 1–28.

[2] Subbu Kannan, Vairaprakash Gurusamy, S Vijayarani, J Ilamathi, M Nithya, S Kannan, and V Gurusamy. 2014. Preprocessing techniques for text mining. *International Journal of Computer Science & Communication Networks* 5, 1 (2014), 7–16.

[3] Aji Nugraha Santosa Kasmaji and Ayu Purwarianti. 2015. Employing natural language processing to analyse grammatical error in a simple Japanese sentence. In *2015 International Conference on Electrical Engineering and Informatics (ICEEI)*. IEEE, 82–86.

[4] Michael Levy. 1997. *Computer-assisted language learning: Context and conceptualization*. Oxford University Press.

[5] Paul McCann. 2020. fugashi, a Tool for Tokenizing Japanese in Python. *arXiv preprint arXiv:2010.06858* (2020).

[6] Melanie Siegel. 1999. The syntactic processing of particles in Japanese spoken language. *arXiv preprint cs/9906003* (1999).