

Research Proposal

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

ACM Reference Format:

Ana Verulidze. 2022. Research Proposal Detecting typographic, particle and sentence formation errors in JLPT N5 sentences using a rules-based system. In *Proceedings of ACM Conference (Conference'17)*. ACM, New York, NY, USA, 4 pages. <https://doi.org/>

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 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 heavily depends on the instructor and cannot be fully performed by a computer at this point [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 "びょういん" (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 omit "私は" (I + particle) from the above mentioned sentence without changing the meaning conveyed.

This 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 accordingly. [3].

3 Background

3.1 A brief introduction to the 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 translate to "I go to school by bike every day":

- "私は (I + particle) 毎日 (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) 毎日 (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) 毎日 (every day) 自転車で (bike + particle) 私は (I + particle)。"

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Conference'17, July 2017, Washington, DC, USA

© 2022 Association for Computing Machinery.

ACM ISBN 978-1-4503-XXXX-X/18/06...\$15.00

<https://doi.org/>

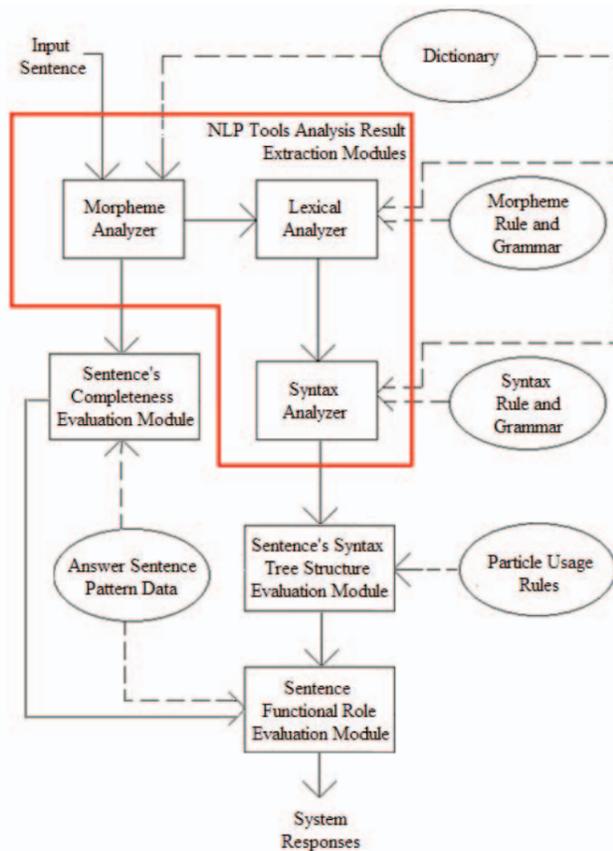


Figure 1: Software architecture of Kasmaji et al's error-detection and handling tool [3]

- "自転車で (bike + particle)行きます (to go) 毎日 (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 complement phrases and arguments, modify verbs and nouns, and have 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 "私は水を飲みます" (I am drinking water) can be shortened as "水を飲みます" (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 text for further analysis. This process usually faces challenges that depend on the type of language used in 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 syntax 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.

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 to be compatible 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, "日曜日に" (on Sunday) "うちで" (at home), and "映画を" (movie) are the children of "見ました" (to watch). Based on this information we can deduce that the input sentence was "日曜日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 "日曜日に", "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). The author describes 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 depends 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].

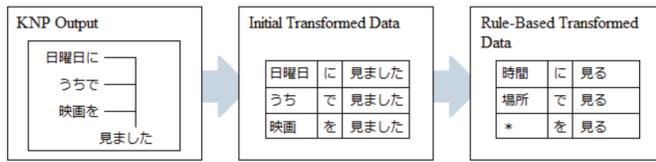


Figure 2: Steps in the Structure evaluation unit [3]

4 Design

The implementation of the error-detecting tool will have preprocessing, analysis and evaluation phases. The preprocessing phase will cover NLP tool implementations including Fugashi [5], Sudachipy, JUMAN++ [7] and KNP [8]. JUMAN++ is an improved version of JUMAN, which was used in Kasmaji et al’s implementation. The analysis phase will be divided into two sections that deal with spelling errors/word omissions and sentence formation/particle usage separately. Evaluation phase will take care of describing errors and possibly proposing improvements.

4.1 Preprocessing Phase

Preprocessing phase will cover two distinct sections: syntax tree production phase and a mapping phase. In the syntax tree production phase, input text will be sent for morphological analysis to JUMAN++, whose output will be modified and sent to KNP, that will produce a syntax tree. In the mapping section, first, Fugashi will be used to tokenize the input text and 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 “図書館で勉強しています” (I am studying at the library) as “図書館 (library) で (particle indicating location) 勉強 (study, noun) し (root of “to do”) て (te-form indicating the tense) い (morpheme) ます (polite form)”. In order to make dictionary look ups simpler, “勉強しています” will be combined into a single token. This will be done through my previously implemented word splitting tool. A visual representation of the preprocessing phase can be seen on Figure 3.

4.2 Analysis Phase

In the analysis phase, similarly to Kasmaji et al’s implementation, KNP syntax tree output will be processed. The child nodes in the tree will be split into words and particles that follow them. After that, 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, we will be comparing the grammatically correct parts of speech with the extracted parts of speech. This will help evaluate how well the student uses particles. At the same time, we will be able to determine whether the sentence structure is valid, by allowing child nodes in the syntax tree to be ordered in any way as long as they are followed with the root. It is possible that sentences may contain more than one verb, in which case multiple syntax trees will be produced for each of the connecting sections within the sentence, and the same rules will be applied separately. For example, the following sentence: “猫の名前をポキにします、この名前がかわいいと思いますから” (I will name my cat Poki, because I

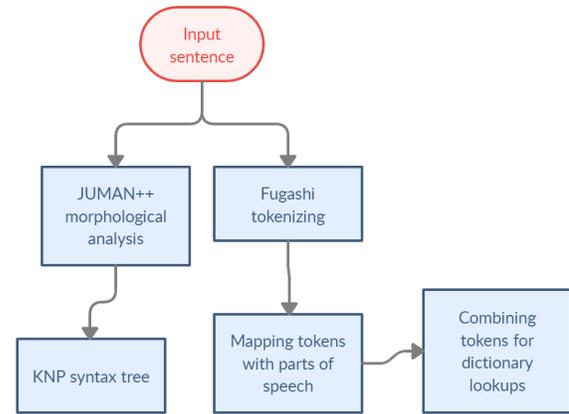


Figure 3: Steps in the Preprocessing Phase

think this is a cute name) will produce two syntax trees with roots “します (to do, in this case - to name) and 생각합니다 (to think)”. In this phase we will also take care of the combined tokens provided by the word splitting tool. These words 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 Analysis phase can be viewed on Figure 4.

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 we can output to the user. The user interface will be created using either Django or Flask, since all of 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 Budget and Risks

This project will use Juman++ [7], KNP [8], Fugashi [5], Sudachipy, Django or Flask, all of which are freely available online, therefore there will be no costs associated with it. The major risk we may encounter will be ensuring the validity of the feedback provided by the tool. Considering that I am still learning Japanese, there is a chance that I will miss some errors. To resolve this issue, I will seek feedback from native speakers and language professors.

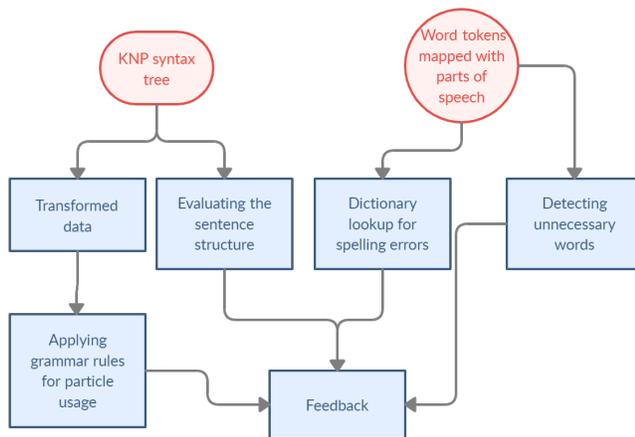


Figure 4: Steps in the analysis phase

6 Timeline

- Week 1 - Create the web interface for the tool. Functionally, the application at this point may only include text boxes and a button which returns the input text without transforming it in any way. The application design aspect should be fairly complete.
- Week 2 - Work on the Preprocessing Phase - syntax tree production using JUMAN++ and KNP
- Week 3 - Work on the Preprocessing Phase - mapping section and combining tokens
- Week 4 - Move on to the Analysis phase for the syntax tree, transform the output to make it easier to read and process
- Week 5 - implement a data structure responsible for applying grammar rules and use it to process the data given by the syntax tree
- Week 6 - Get the first version of the tool up on the web application. This version will not provide feedback, but it should list sentence structure and particle usage related errors
- Week 7 - Continue working on the analysis phase, focusing on spelling errors and detecting unnecessary words
- Week 8 - Combine all previous work, test and debug
- Week 9 - Move on to the evaluation phase: combine results into a feedback string and display it on the web application
- Week 10 - User testing. Have Japanese language students test the tool and provide feedback
- Week 11 - Work on the paper and the feedback provided by the students
- Week 12 - Continue working on the paper, finalize the error-detection tool
- Week 13 - Second phase of user testing. Work on the feedback provided
- Week 14 - finalize and polish

7 Acknowledgement

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

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).
- [7] Arseny Tolmachev, Daisuke Kawahara, and Sadao Kurohashi. 2020. Design and structure of the Juman++ morphological analyzer toolkit. *Journal of Natural Language Processing* 27, 1 (2020), 89–132.
- [8] Arseny Tolmachev, Hajime Morita, and Sadao Kurohashi. 2016. A grammar and dependency aware search system for Japanese sentences. *Corpus* 1, 2 (2016), 3–4.