GlyphFormer: Improving Japanese Language Models with Sub-character Tokenization

Koichiro Kanno
artisan.baggio@gmail.com

August 18, 2024

Abstract

In this study, we investigate the effectiveness of sub-character tokenization in Japanese language processing using the ALBERT model [1]. We focus on radical-based and element-based sub-character tokenization and compare these methods with traditional character-based tokenization. Evaluations were conducted on a dataset of 500 sentences extracted from the Japanese novel "Botchan." The results demonstrate that employing radical and element-based approaches significantly improves the model's perplexity. Further experiments reveal that this improvement is not merely due to increased data volume from tokenization but is attributed to the inherent benefits of decomposing kanji characters.

1. Introduction

Research on Transformer models in natural language processing (NLP) has predominantly focused on alphabet-based languages, with limited studies on multibyte character languages like Japanese. Unlike English, where words are clearly separated by spaces, Japanese text lacks explicit delimiters, making word segmentation a complex task. Therefore, tools like MeCab [2] and SentencePiece [3] are commonly used to tokenize Japanese text into words or subwords before applying Transformer models.

Japanese text heavily utilizes kanji characters, which consist of various components conveying semantic and phonetic information. This study explores how decomposing kanji into smaller units—sub-characters—affects the performance of the ALBERT model in Japanese language processing.

Kanji characters are classified into four types based on their formation:

- **Pictographs:** Characters that are visual representations of objects (e.g., 山 (mountain), 川 (river)).
- **Ideographs:** Characters that represent abstract concepts through lines or dots (e.g., 上 (up), 下 (down)).
- **Compound Ideographs:** Characters formed by combining multiple characters to convey a new meaning (e.g., 林 (forest), composed of two 木 (tree) characters).
- **Phono-semantic Characters:** Characters formed by combining a phonetic component with a semantic component, known as a radical (e.g., 語 (word), where 言 (speech) is the radical).

The Formation of the Kanji "親"

Kanji characters have a rich history, evolving in form and meaning over time [4]. The kanji 親 (parent) is a prime example of a Compound Ideograph. It combines multiple components, each carrying its own meaning, to convey a new, composite meaning.

- **Composition of "親"**: The character 親 is composed of 立 (stand), 木 (tree), and 見 (see). According to one theory, these components together symbolize a person standing by a tree watching or nurturing, metaphorically representing "parent."

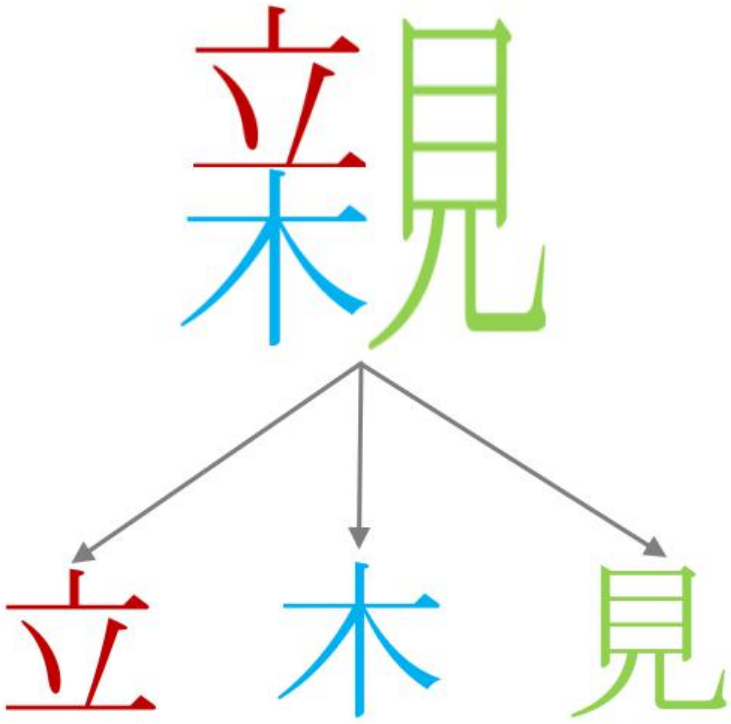

Figure 1 : The Formation of the Kanji "親"

This historical evolution of kanji is also reflected in the changes in their script forms over time:

- **Oracle Bone Script (甲骨文字)**: The oldest form of Chinese characters, inscribed on turtle shells and animal bones.
- **Bronze Script (金文)**: Characters inscribed on bronze vessels, featuring more complex shapes.
- **Small Seal Script (小篆)**: Standardized characters under Emperor Qin Shi Huang, forming the basis of modern kanji.
- **Regular Script (楷書)**: The modern form of kanji used today.

Understanding both the types of kanji and their historical evolution provides valuable context for decomposing kanji into sub-characters.

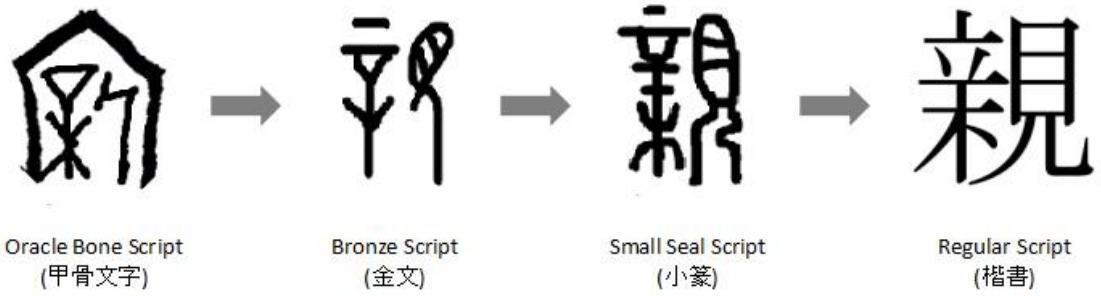

Figure 2 : The Evolution of the Kanji "親"

2. Related Work

A previous study using LSTM models demonstrated that breaking down kanji into sub-characters could improve model performance on Japanese NLP tasks [4]. However, this study did not explore the use of more advanced Transformer-based models like ALBERT. My research aims to extend these findings by applying sub-character tokenization techniques to the ALBERT model.

---

3. Methodology

The experiments were conducted using the ALBERT model, focusing on two sub-character tokenization methods: radical-based and element-based [5]. These methods were compared against traditional character-based tokenization using a dataset derived from the novel "Botchan," which contains 500 sentences out of a total of 2715.

3.1 Visualization of Kanji "親" Decomposition

The following images illustrate the different methods of decomposing the kanji 親:

- **Character-based:** Treating 親 as a single character.
- **Radical-based:** Decomposing 親 into radicals 立 (stand), 木 (tree), and 見 (see), which constitute the main components of the character.
- **Element-based:** Further decomposing 親 into elements 立, 木, 見, 亠 (lid), 目 (eye), including both radicals and finer components.

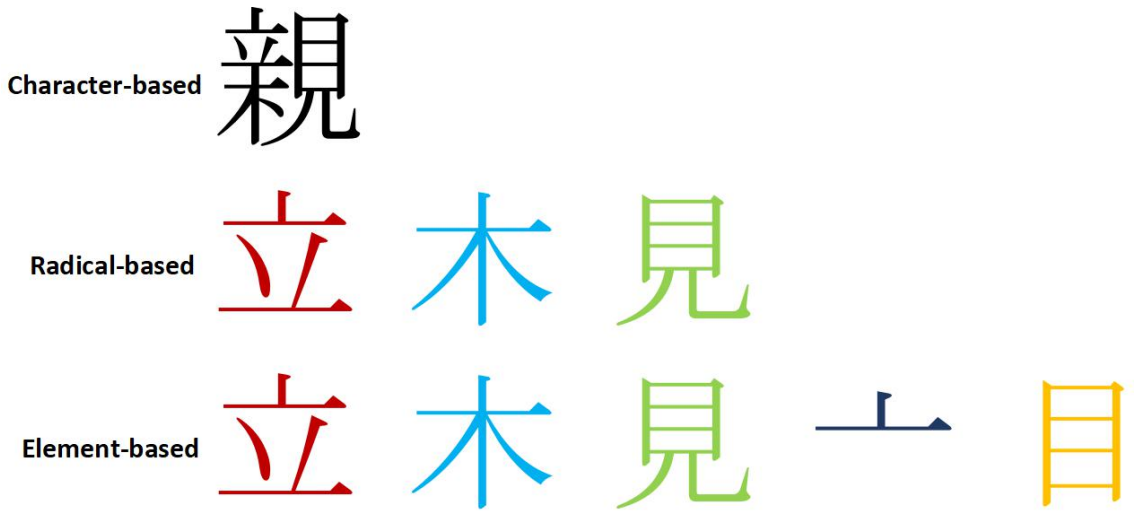

Figure 3 : The Decomposition of the Kanji "親"

3.2 Tokenization Examples

First, we used MeCab [2], a morphological analyzer, to tokenize the text into subwords. MeCab is widely used in Japanese NLP for segmenting text into morphemes, the smallest units of meaning.

The sentence "親譲りの無鉄砲で小供の時から損ばかりしている。" was tokenized as follows:

- **Character-based (MeCab)**: 親譲り の 無鉄砲 で 小 供 の 時 から 損 ばかり し て いる 。
- **Radical-based (Radical)**: 立 木 見 言 襄 り の 丿 一 灬 金 失 石 包 で 小 亻 共 の 日 寺 か ら ま 員 ば か り し て い る 。
- **Element-based (Element)**: 立 木 見 亠 目 言 六 衣 八 一 亠 口 襄 三 り の 灬 一 丿 金 大 夫 失 丿 己 包 石 丿 口 勹 で 小 八 共 亻 の 日 寸 土 寺 か ら 員 口 目 貝 ま ば か り し て い る 。

3.3 Model Training

The model was trained using the following parameters:

- Model: ALBERT without pre-training
- Learning rate: 5 × 10^{-5}
- Optimizer: AdamW

Perplexity was used as the evaluation metric, calculated as:

$$\text{Perplexity} = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\ln P(w_i)\right)$$

Where N is the number of words in the dataset, and $P(w_i)$ is the probability of word w_i.

---

4. Results

4.1 First Phase of Evaluation

In the initial evaluation, the model was trained using the MeCab, Radical, and Element tokenization methods. The results are presented in Figure 4 and summarized in Table 1.

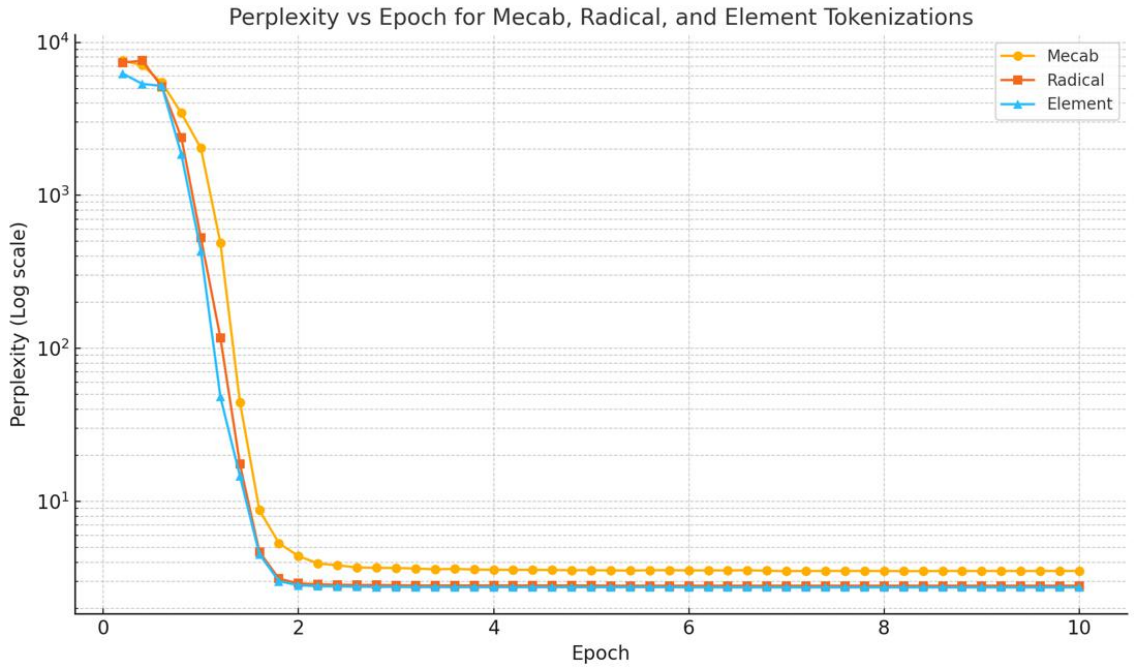

Figure 4 : Perplexity Over Epochs for MeCab (Standard), Radical, and Element Tokenization Methods

Table1: Perplexity and Improvement Rates in the First Phase

Tokenization Method	Perplexity	Improvement (%)
MeCab (Standard)	3.51	-
Radical	2.81	19.94
Element	2.74	21.94

Improvement rates were calculated based on the perplexity of MeCab (Standard).

4.2 Second Phase of Evaluation

Since radical-based and element-based tokenizations increase the number of tokens, leading to more training data, we questioned whether the performance improvement was merely due to the increased data volume. To investigate this, we adjusted the number of tokens in MeCab to match that of the Element method, termed "MeCab (Adjusted to Element Token Count)," and conducted a second evaluation. The perplexity values across epochs for all tokenization methods are shown in Figure 5.

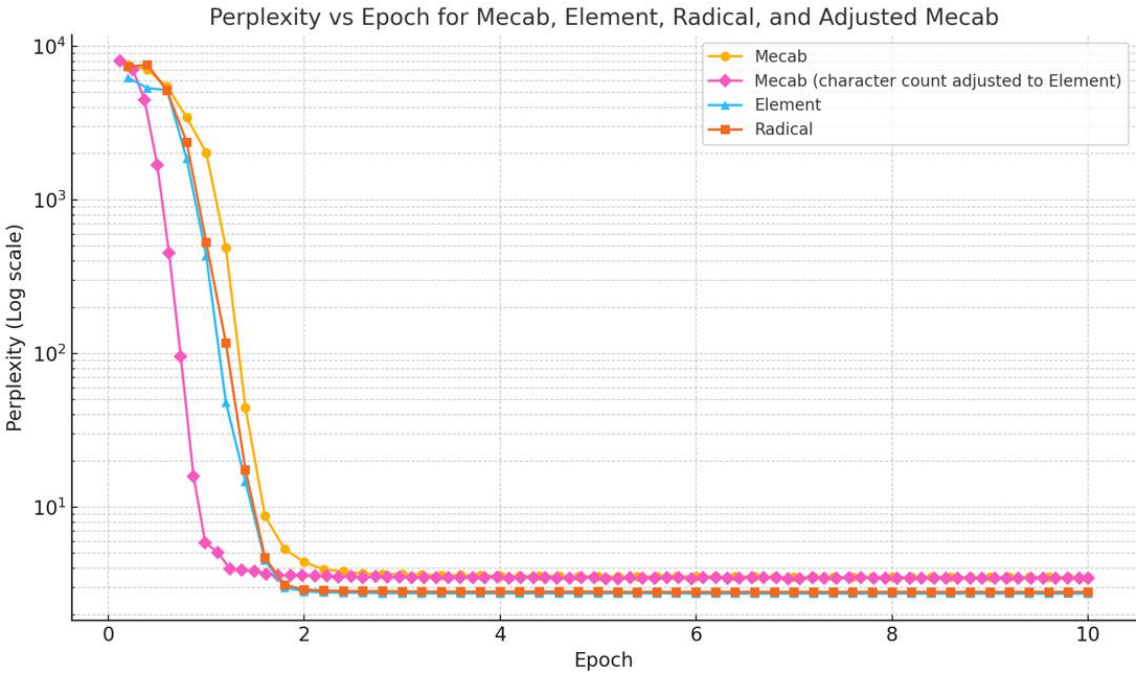

Figure 5 : Perplexity Over Epochs Including MeCab (Adjusted to Element Token Count)

Table 2: Perplexity and Improvement Rates in the Second Phase

Tokenization Method	Perplexity	Improvement (%)
MeCab (Standard)	3.51	-
MeCab (Adjusted to Element Token Count)	3.48	0.85
Radical	2.81	19.94
Element	2.74	21.94

Improvement rates were calculated based on the perplexity of MeCab (Standard).

---

5. Discussion

In the first phase, radical-based and element-based tokenizations showed approximately a 20% improvement in perplexity compared to MeCab (Standard). However, this raised the question of whether the improvement was simply due to the increased data volume from the higher token counts.

In the second phase, we adjusted the MeCab token count to match that of the Element method, creating MeCab (Adjusted to Element Token Count). While MeCab (Adjusted) began to decrease in perplexity earlier than the other methods, it ultimately achieved only about a 0.85% reduction in perplexity, and the final perplexity remained almost unchanged compared to MeCab (Standard). Meanwhile, the Element method continued to exhibit the lowest perplexity.

These results indicate that the performance improvement cannot be solely attributed to increased data volume. Therefore, we conclude that sub-character tokenization by decomposing kanji into radicals and elements inherently contributes to enhancing model performance.

---

6. Conclusion

In this study, we demonstrated that radical-based and element-based sub-character tokenization improves the performance of the ALBERT model in Japanese NLP tasks. Specifically, incorporating sub-character information through kanji decomposition plays a significant role in model learning.

Future work will involve applying these findings to larger datasets and various NLP tasks. Further investigation is needed to identify the optimal sub-character tokenization methods based on kanji types and linguistic tasks.

Additionally, these methods may be applicable not only to Japanese but also to other languages such as Chinese, Arabic, and potentially English.

For more details on the tools and implementations used in this research, please refer to the GlyphFormer project on GitHub [7].

References

[1] ALBERT details on HuggingFace: https://huggingface.co/albert/albert-base-v2

[2] MeCab official site: http://taku910.github.io/mecab/

[3] SentencePiece GitHub repository: https://github.com/google/sentencepiece

[4] Reference on the Formation of 親: https://asia-allinone.blogspot.com/2021/09/p897.html

[5] V. Nguyen, J. Brooke, and T. Baldwin, "Sub-character Neural Language Modelling in Japanese," in Proc. of the First Workshop on Subword and Character Level Models in NLP, Copenhagen, Denmark, Sep. 2017, pp. 148-153. https://aclanthology.org/W17-4122.pdf

[6] KanjiVG-radical: https://github.com/yagays/kanjivg-radical/tree/master

[7] GlyphFormer Project on GitHub: https://github.com/artisanbaggio/GlyphFormer/tree/main