1L1-5

Combining Multiple Dictionaries to Improve Tokenization of Ainu Language

Michal Ptaszynski*1 Yuka Ito*2 Karol Nowakowski*3 Hirotoshi Honma*2
Yoko Nakajima*2 Fumito Masui*1

*1 Department of Computer Science, Kitami Institute of Technology
*2 Department of Information Engineering, National Institute of Technology, Kushiro College
*3 Independent Researcher

In this paper we present our research in improving a tokenizer for Ainu language. Tokenization is a process where a sentence is separated into basic elements, such as words or morphemes. Ainu language is a critically endangered language of Ainu people living in northern parts of Japan. Since Ainu language originally did not have a writing system, documents in Ainu language are usually transcribed in an unsystematic way. To allow effective processing and contribute to further revitalization of Ainu language, we combine multiple official Ainu language dictionaries to improve tokenization of such documents. We also compare state-of-the-art tokenizer with custom one based for the needs of this research.

1. Introduction

Ainu language is a critically endangered language isolate spoken by the Ainu people living on northern parts of Japan, such as Aomori and Hokkaido, extending to Sakhalin. Although Ainu reside in parts of the world inhabited mostly by Japanese, Chinese (modern Mongoloids), or Native American-related Inuits, they are genetically unrelated to those peoples [1]. Their language has also remained unique in its kind, with no proof of origin or relation to any other known world language [2]. The critical situation of the language is revealed by the fact that although the population of Ainu-related people is estimated on around 25 thousand [3], the number of Ainu community members capable of speaking Ainu language fluently is estimated to less than hundred [4].

To contribute to the task of preserving and revitalizing of the Ainu language, Ptaszynski et al. [5] proposed **POST-AL**, or ***P**art **o**f **S**peech **T**agger for **A**inu **L**anguage*, which included a tokenizer, part-of-speech tagger, and a simple word-to-word translator for Ainu language. However, there have been some tasks remained in the POST-AL system. The most crucial one was the limited dictionary base, which hindered all parts of the system. As Ptaszynski et al. [11] later confirmed, various dictionaries influence the performance of the system in different ways. Therefore, in this research we focused on combining several of the existing dictionaries to improve the Ainu language tokenizer, as the first step in improving the general performance of the whole POST-AL system. We also compared the proposed method to the state-of-the-art, namely, the tokenizer included in Natural Language Toolkit (NLTK), trained on the provided Ainu language data.

In the following sections we firstly present the methods applied in the research, namely, POST-AL system as a whole with the Ainu language tokenizer in particular, and the NLTK tokenizer. We also describe all dictionaries used to create various dictionary base combinations, and perform two evaluation experiments. First, to find out which tokenizer works better for Ainu language, and second, to chose the optimal dictionary base combination. Finally, we conclude the paper and discuss further improvements.

Contact: Michal Ptaszynski, Kitami Institute of Technology, Koencho 165, 090-8507, Kitami, Japan, Tel./Fax:0157-26-9327, E-mail: ptaszynski@cs.kitami-it.ac.jp

2. Tokenizers

2.1 POST-AL Tokenizer

The proposed tokenizer is a part of Ainu language toolkit developed by Ptaszynski et al. [6], which consists of the mentioned tokenizer, POST-AL, or ***P**art **o**f **S**peech **T**agger for **A**inu **L**anguage* [5], basic word-to-word Ainu to Japanese translator, and a simple chunker. The toolkit was originally using a dictionary base created on the "Lexicon to Yukie Chiri's Ainu Shin-yōsyū (Ainu Songs of Gods)" created originally by Kirikae (2003) [7].

The included tokenizer is based on a standard approach to tokenization, namely, dictionary lookup, with further modifications, such as matching beginning from the longest string in lexicon (longest string matching). The tokenizer also keeps track of the already matched word patterns to avoid over-tokenization, or splitting each word recursively down to its smallest elements.

2.2 NLTK Word Tokenizer

For performance comparison we used Word Tokenizer included in the NLTK, or Natural Language Tool-Kit*1. However, since originally the tokenizer was based for English, we re-trained it on the same dictionary base as included in POST-AL.

3. Dictionaries

3.1 Lexicon to Ainu Songs of Gods

As the first dictionary we applied the one used originally in POST-AL, namely, *Ainu shin-yōshū jiten* (Lexicon to Yukie Chiri's Ainu Shin-yōsyū (Ainu Songs of Gods)) by Kirikae (2003) [7] (later abbreviated to KK). It is one of the newest Ainu language dictionaries with a part-of-speech classification standard developed to highlight the specificities of parts of speech in Ainu language. Thus, except traditional POS names like nouns or verbs, the dictionary contains parts of speech either rare or not existing in other languages, such as "interrogative indefinite adverb", like *hempara*, "demonstrative adverbs", like *ene* or *neno*, or "postpositive adverb", like *ari, epitta*.

*1 http://www.nltk.org/

The dictionary contains 2,019 entries, each containing five types of information: token (word, morpheme, etc.), part of speech (POS), meaning (in Japanese), reference to the traditional Ainu story (*yukar*) it appears in, and usage examples (not for all cases).

3.2 Ainu Conversational Dictionary

Ainugo kaiwa jiten (Ainu conversational dictionary) [8] is one of the first dictionaries for Ainu language collected by a Japanese researcher. Shōzaburo Kanazawa, with help of Kotora Jinbo, collected it firstly around 1895 and 1897, right after Piłsudski's first collection [9], and a few years before Batchelor published his first Ainu-English-Japanese dictionary [10]. The most recent reprint of the dictionary is dated on 1986.

In its original form, the dictionary contains 3,839 entries. However, for the need of this research we have adopted the version developed by Ptaszynski et al. [11], where they divided the original multi-word entries and created 12,855 separate one-word entries, which helped increase the Recall rate when the dictionary was applied in POS tagging. Later we refer to this dictionary as JK, after the initials of its original creators.

3.3 Combined Dictionary

Finally, we combined the two above dictionaries. Cases where the same word appeared in both dictionaries were automatically unified based on their Japanese translations. The final combined dictionary contained 4,161 entries. In the following sections we will refer to this dictionary as JK+KK.

4. Datasets

4.1 *Yukar* - Ainu Songs of Gods

As the dataset for evaluation we used a collection of 13 Ainu stories (*yukar*) included in *Ainu shin-yōshū* (Ainu Songs of Gods) gathered by Chiri (1978) [12]. The stories have been partially processed by Kirikae [7], who separated them into words (tokenized) manually according to linguistic rules (the original texts were transcribed and separated by Chiri according to poetic rules). Therefore this set is ideal for evaluating the tokenization performance. Later we abbreviate these materials to "Yuk."

4.2 Shibatani's Colloquial Text Samples

Although *Ainu Songs of Gods* is a sufficiently good source for testing tokenization performance, the original dictionary base was created on the same data, meaning, that performance when tested with this dictionary would be higher. Therefore we decided to apply other datasets as well. As the first one we used colloquial text samples included in Shibatani's *The Languages of Japan* [2]. The book, among others, contains sixteen examples of longer sentences in Ainu language, tokenized according to linguistic rules, similarly to the work done by Kirikae (2003) [7]. Later we abbreviate these materials to "Shib."

4.3 Mukawa Dialect Samples

We also used "The Japanese-Ainu Dictionary of Mukawa Dialect with Sound" [13], containing 20,147 of various sentence samples. It is a dictionary developed on the basis of 150 hours of speech in Ainu language (Mukawa dialect), recorded by Tatsumine Katayama. The dictionary was developed by Hiroshi Nakagawa and Yuko Honda under the grant "Descriptive Study of Mukawa Dialect of Ainu using Recorded Materials." Later we abbreviate these materials to "Muk."

Table 1: Results of comparison between two applied tokenizers.

	POST-AL tokenizer			**NLTK word tokenizer**		
	Pr	**Re**	**F1**	**Pr**	**Re**	**F1**
Yuk09	81.3%	85.9%	83.2%	38.4%	92.9%	53.8%
Yuk10	90.2%	93.4%	91.5%	28.8%	82.1%	42.1%
Yuk11	86.0%	90.6%	87.9%	31.9%	89.1%	46.5%
Yuk12	84.5%	87.7%	85.7%	33.1%	86.0%	46.9%
Yuk13	87.4%	92.7%	89.6%	34.8%	86.1%	49.0%

5. Evaluation Experiments

We performed two evaluation experiments. In the first one, we compared the two tokenizers mentioned in section 2. on the same data, namely, *Yukar* - Songs of Gods, in particular the 9th to 13th *yukar* collected by Chiri [12]. In the second experiment we used the best tokenizer from the first experiment and verified its performance on the data mentioned in section 4., by using various dictionary combinations, mentioned in section 3..

5.1 Experiment 1: Comparing Tokenizers

In the comparison of the two applied tokenizers, POST-AL tokenizer and NLTK tokenizer, we used the same dictionary base, namely Kirikae's lexicon [7], and tested it on the five *yukar* stories, from yukar 9 to yukar 13 [12]. Moreover, since the original transcription of Ainu language used by Chiri differs from the present standard (see [7] for details), the data was further preprocessed in the following way. For the needs of experiment we used transcripts of yukar stories provided by Kirikae [7], which follow modern standard in Ainu language transcription. Furthermore, since yukar are poems, traditionally, in transcription they have been divided into short chunks representing pauses in recitation. To make the task more linguistically based, we also re-joined the chunks and re-separated them into actual sentences.

All results were calculated with the means of Precision (P), Recall (R) and balanced F-score (F). Tokenization is in short a process of separating sentence into tokens (words, punctuation marks, etc.), which in practice is realized by separating the sentence by, e.g., spaces. Thus Precision is calculated as the percentage of how many separations, or "spaces", within those proposed by the system were correct. It is calculated as in equation 1. Recall is the percentage showing how many correct spaces the system proposed within all possible correct spaces. It is calculated as in equation 2. The balanced F-score is a harmonic mean of the two values. It is calculated as in equation 3. All results were represented in Table 1. Since both the dictionary and test data were of similar origin (yukar), it was predictable that the results should be high, and the only differences would come from the tokenizer algorithm itself.

As for the results, NLTK tokenizer, although obtaining comparable Recall to POST-AL tokenizer, reached much lower Precision. The analysis of errors showed, that NLTK tokenizer re-trained on the new dictionary base, recursively divided each separated word, thus suffering from over-tokenization. This problem is dealt with in POST-AL tokenizer by keeping track of separated words.

However, since the results of POST-AL tokenizer were still not ideal we plan to further improve the tokenization by, e.g., applying contextual information (usage examples from the dictionary, etc.).

Table 2: Results of all dictionaries and their combinations, under all evaluation conditions.

		Yuk9	Yuk10	Yuk11	Yuk12	Yuk13	JK samples	Shib	Muk	Ave.	Input text version:
JK	Precision	53.8%	57.7%	56.2%	50.4%	58.3%	86.2%	64.9%	69.0%	62.1%	
	Recall	53.8%	57.7%	56.2%	50.4%	58.3%	86.2%	64.9%	69.0%	62.1%	Postprocessed
	F-score	53.8%	57.7%	56.2%	50.4%	58.3%	86.2%	64.9%	69.0%	62.1%	(no spaces)
	Precision	55.4%	60.8%	60.1%	51.7%	65.3%	79.9%	n/a	n/a	62.2%	
	Recall	55.4%	60.8%	60.1%	51.7%	65.3%	79.9%	n/a	n/a	62.2%	Original
	F-score	55.4%	60.8%	60.1%	51.7%	65.3%	79.9%	n/a	n/a	62.2%	
	Precision	53.2%	58.2%	56.0%	49.1%	57.3%	73.6%	n/a	n/a	57.9%	
	Recall	53.2%	58.2%	56.0%	49.1%	57.3%	73.6%	n/a	n/a	57.9%	Original
	F-score	53.2%	58.2%	56.0%	49.1%	57.3%	73.6%	n/a	n/a	57.9%	(no spaces)
KK	Precision	81.6%	83.1%	91.1%	77.4%	87.0%	66.8%	57.8%	65.5%	76.3%	
	Recall	81.6%	83.1%	91.1%	77.4%	87.0%	66.8%	57.8%	65.5%	76.3%	Postprocessed
	F-score	81.6%	83.1%	91.1%	77.4%	87.0%	66.8%	57.8%	65.5%	76.3%	(no spaces)
	Precision	83.9%	86.8%	87.1%	76.9%	87.7%	68.5%	n/a	n/a	81.8%	
	Recall	83.9%	86.8%	87.1%	76.9%	87.7%	68.5%	n/a	n/a	81.8%	Original
	F-score	83.9%	86.8%	87.1%	76.9%	87.7%	68.5%	n/a	n/a	81.8%	
	Precision	79.4%	82.0%	87.4%	74.6%	85.3%	56.3%	n/a	n/a	77.5%	
	Recall	79.4%	82.0%	87.4%	74.6%	85.3%	56.3%	n/a	n/a	77.5%	Original
	F-score	79.4%	82.0%	87.4%	74.6%	85.3%	56.3%	n/a	n/a	77.5%	(no spaces)
JK+KK	Precision	73.4%	80.4%	81.9%	73.9%	85.3%	82.9%	70.8%	69.0%	77.2%	
	Recall	73.4%	80.4%	81.9%	73.9%	85.3%	82.9%	70.8%	69.0%	77.2%	Postprocessed
	F-score	73.4%	80.4%	81.9%	73.9%	85.3%	82.9%	70.8%	69.0%	77.2%	(no spaces)
	Precision	78.8%	84.1%	80.8%	74.6%	85.7%	78.5%	n/a	n/a	80.4%	
	Recall	78.8%	84.1%	80.8%	74.6%	85.7%	78.5%	n/a	n/a	80.4%	Original
	F-score	78.8%	84.1%	80.8%	74.6%	85.7%	78.5%	n/a	n/a	80.4%	
	Precision	70.9%	79.9%	78.4%	71.1%	80.7%	69.9%	n/a	n/a	75.2%	
	Recall	70.9%	79.9%	78.4%	71.1%	80.7%	69.9%	n/a	n/a	75.2%	Original
	F-score	70.9%	79.9%	78.4%	71.1%	80.7%	69.9%	n/a	n/a	75.2%	(no spaces)

$$P = \frac{correctly\ \ predicted\ \ spaces}{all\ \ proposed\ \ spaces} \quad (1)$$

$$R = \frac{correct\ \ predicted\ \ spaces}{all\ \ gold\ \ standard\ \ spaces} \quad (2)$$

$$F_1 = 2\frac{P * R}{P + R} \quad (3)$$

5.2 Experiment 2: Combining Dictionaries

Experiment Setup: Since the first experiment revealed that the custom tokenizer built especially for the Ainu language obtained much higher results, in the next experiment we applied the winning POST-AL tokenizer. The goal of the second experiment was to verify what is the optimal dictionary base for the tokenizer. We compared Kirikae's lexicon [7] (KK), Kanazawa-Jinbo dictionary [8] (JK), and the combination the two (JK+KK).

As the first test data we used the same four yukar stories as in the first experiment. We also verified the performance under several conditions. Firstly, as it was mentioned in section 5.1, The original data [12] does not follow the modern standard of Ainu language transcription. However, we can predict that in practice, the tokenizer sometimes will be used to tokenize also texts in non-standard transcription. Therefore we verified the performance both for the original transcription used by Chiri [12] (Input text version: Original), as well the one postprocessed later by Kirikae [7] (Input text version: Postprocessed). This time however, for practical reasons, we used the line-splitting representing recitation pauses (verses), and not full sentences, as int he first experiment (thus differences in KK performance for yukar 9-13). Finally, originally, verses also included spaces in the middle of the line representing a caesura (a break in a verse indicating a separation of two recited phrases). In the experiment we checked if caesura helped or hindered the performance by using the dataset either with caesurae (Original), or without (Original (no spaces)).

However, since the performance of the system could be influenced by the data it was trained on. Therefore, we also used sentence examples from Ainu Conversational Dictionary [8], to check if indeed the system will perform better on samples containing its training data.

To also verify the tokenizer in more objective conditions, we also used data unrelated to training data, namely, text samples from Shibatani [2] (Shib) and sentence samples from Mukawa Dialect Dictionary [13].

Results: The results for all verification conditions were represented in Table 2.

First of all, the tokenizer usually worked best on data properly transcribed (Postprocessed). However, performance on non-standard transcription was also sufficient, and in some cases better, which means the system can be used also in other transcription

Table 3: Analysis of tokenization errors.

Tokenizer	Gold std	Category
kutun kutun	kutunkutun	Dictionary
a sawa	as a wa	Tokenizer
tasi ne	ta sine	Tokenizer
karku su	kar kusu	Tokenizer
neap	ne a p	Tokenizer
aw a	a wa	Tokenizer
kuru n	kur un	Tokenizer
nep	ne p	Tokenizer
cir uska	ci ruska	Tokenizer
ciki k	ci kik	Tokenizer
cioarkaye	ci oarkaye	Tokenizer
ayke	a ike	Tokenizer
pokna sir	poknasir	Dictionary
ciousi	ci ousi	Tokenizer
montum	mon tum	Tokenizer
cioarkaye	ci oarkaye	Tokenizer
petetok o	pet etoko	Tokenizer
pirkare ra	pirka rera	Tokenizer
isoytak	isoitak	Dictionary

schemas.

We also found out that caesura always helps in tokenization. It provides an additional hint in tokenization, and when deleted often causes errors.

As predicted, tokenization based on Kirikae's lexicon achieved higher results for yukar stories, while Kanazawa-Jinbo's dictionary was better for processing sentence samples from that dictionary. However, the performance for the combined dictionary (JK+KK) did not drop severely. Moreover, what is important, the combined dictionary performed much better on data unrelated to training data, and in average.

Error Analysis: To verify what was the most common cause of errors we looked at wrongly tokenized words from yukar 10. The results we represented in Table 3. Most common cause of errors was the fault in the tokenization process (see "Tokenizer" in the far right column of Table 3). Some of the errors were caused by the structure of the dictionary. For example, words in lexicon are sorted alphabetically. Moreover, the tokenizer also looks up words starting from the longest (longest match principle). Thus, e.g., phrase *awa* was divided into *aw a*, and not into *a wa*. This could be improved if the dictionary lookup also included other information, such as statistical probability of occurrence of a word, or contextual information.

However, there were also errors unrelated to the tokenizer, but which resulted in lacks in dictionary base (e.g., lack of word *poknasir*), or minor inconsistencies in transcription (*isoytak* vs. *isoitak*). Such errors can be easily corrected by further improving the dictionary base. Some improvement could also be achieved by adding an external layer for transcription correction.

6. Conclusions and Future Work

The paper presented our research in improving a tokenizer for Ainu language, an endangered language of Ainu people living in northern Japan (e.g., Hokkaido). Tokenization is a process where a sentence is separated into basic elements, such as words or morphemes, and is the first step in other NLP tasks, such as part-of-speech tagging, parsing, etc.

With a goal to revitalize Ainu language, in previous research we proposed Ainu language toolkit [6]. At first, we verified that the proposed tokenizer, which is part of the toolkit, works better for Ainu language than other state-of-the-art tokenizer. To improve the proposed tokenizer we combined two official Ainu language dictionaries. We found out that the combination improved performance on objective samples unrelated to training data.

In the future we plan to include in the combinations also other dictionaries, such as the one by Nakagawa (1995) [14], or Tamura (1998) [15]. We also plan to improve the tokenizer by adding to the equation statistical probability of occurrence of a word (e.g., term frequency), and contextual information (surroundings of a word).

References

[1] Margaret Sleeboom. 2004. *Academic Nations in China and Japan*. Routledge: UK.

[2] Masayoshi Shibatani. 1990. *The Languages of Japan*. Cambridge university Press, London.

[3] Poisson, B. 2002. *The Ainu of Japan*. Lerner Publications, Minneapolis, p. 5.

[4] Skye Hohmann. 2008. The Ainu's modern struggle. In *World Watch*, Vol 21., No. 6, pp. 20 ・ 4.

[5] Michal Ptaszynski and Yoshio Momouchi. 2012. Part-of-Speech Tagger for Ainu Language Based on Higher Order Hidden Markov Model, *Expert Systems With Applications*, Vol. 39, Issue 14 (2012), pp. 11576-11582.

[6] Michal Ptaszynski, Fumito Masui and Yoshio Momouchi. 2013. A Toolkit for Analysis of Ainu Language, In Demo Session of *20th International Conference on Language Processing and Intelligent Information Systems (LP&IIS 2013)*.

[7] Hideo Kirikae. 2003. *Ainu shin-yōshū jiten: tekisuto bumpō kaisetsu tsuki* (Lexicon to Yukie Chiri's Ainu Shin-yōsyū (Ainu Songs of Gods) with Text and Grammatital Notes) [In Japanese]. Published by Daigaku Shorin.

[8] Kotora Jinbo and Shouzaburo Kanazawa. 1986. *Ainugo kaiwa jiten* (Ainu conversational dictionary) [In Japanese]. Sapporo: *Hokkaido publication project center*. First edition: 1898, Tokyo: Kinkōdo.

[9] Bronisław Piłsudski (Author), Alfred F. Majewicz (Editor). 2004. *The Collected Works of Bronislaw Pilsudski: Materials for the Study of the Ainu Language and Folklore*, v.3, Pt. 2: Materials for the Study of the Ainu, (Trends in Linguistics: Documentation). Mouton de Gruyter (Oct 2004)

[10] John Batchelor. 1905. *An Ainu-English-Japanese dictionary (including a grammar of the Ainu language)*. Tokyo Methodist Pub. House.

[11] Michal Ptaszynski, Karol Nowakowski, Yoshio Momouchi, Fumito Masui. 2016. Comparing Multiple Dictionaries to Improve Part-of-Speech Tagging of Ainu Language. In *Proceedings of The 22nd Annual Meeting of The Association for Natural Language Processing (NLP-2016)*, pp. 973-976.

[12] Yukie Chiri. 1978. *Ainu shin-yōshū*. Tokyo, Iwanami Shoten.

[13] The Japanese-Ainu Dictionary of Mukawa Dialect with Sound, http://cas-chiba.net/Ainu-archives/mukawa/

[14] Hiroshi Nakagawa. 1995. *Ainugo Chitose Hōgen Jiten: The Ainu-Japanese Dictionary: Chitose Dialect* [In Japanese]. Sōfūkan.

[15] Suzuko Tamura. 1998. *Ainugo Chitose Hōgen Jiten: The Ainu-Japanese Dictionary: Saru Dialect* [In Japanese]. Sōfūkan.