information

MDPI

Article

MiNgMatch—A Fast N-gram Model for Word Segmentation of the Ainu Language

Karol Nowakowski *, Michal Ptaszynski and Fumito Masui

Department of Computer Science, Kitami Institute of Technology, 165 Koen-cho, Kitami, Hokkaido 090-8507, Japan; ptaszynski@cs.kitami-it.ac.jp (M.P.); f-masui@mail.kitami-it.ac.jp (F.M.)

* Correspondence: karol_nowakowski@interia.pl

Received: 12 September 2019; Accepted: 11 October 2019; Published: 16 October 2019

Abstract: Word segmentation is an essential task in automatic language processing for languages where there are no explicit word boundary markers, or where space-delimited orthographic words are too coarse-grained. In this paper we introduce the MiNgMatch Segmenter—a fast word segmentation algorithm, which reduces the problem of identifying word boundaries to finding the shortest sequence of lexical n-grams matching the input text. In order to validate our method in a low-resource scenario involving extremely sparse data, we tested it with a small corpus of text in the critically endangered language of the Ainu people living in northern parts of Japan. Furthermore, we performed a series of experiments comparing our algorithm with systems utilizing state-of-the-art lexical n-gram-based language modelling techniques (namely, Stupid Backoff model and a model with modified Kneser-Ney smoothing), as well as a neural model performing word segmentation as character sequence labelling. The experimental results we obtained demonstrate the high performance of our algorithm, comparable with the other best-performing models. Given its low computational cost and competitive results, we believe that the proposed approach could be extended to other languages, and possibly also to other Natural Language Processing tasks, such as speech recognition.

Keywords: word segmentation; tokenization; language modelling; n-gram models; Ainu language; endangered languages; under-resourced languages

1. Introduction

One way to handle ambiguity—a major challenge in any Natural Language Processing task—is to consider the target text in context. A typical approach is to use an n-gram model, where the probability of a word depends on the $n-1$ previous words. In this paper we argue that in the context of word segmentation, the problem can be reduced to finding the shortest sequence of n-grams matching the input text, with little or no drop in performance compared to state-of-the-art methodologies. In order to verify the usability of our approach in a scenario involving extremely sparse data, where its performance is expected to suffer the most, we tested it with a small corpus of text in Ainu, a critically endangered language isolate native to the island of Hokkaido in northern Japan.

Word segmentation is a part of the process of tokenization, a preprocessing stage present in a wide range of higher level Natural Language Processing tasks (such as part-of-speech tagging, entity recognition and machine translation), where the text is divided into basic meaningful units (referred to as *tokens*), such as words and punctuation marks. In the case of writing systems using explicit word delimiters (e.g., whitespaces), tokenization is considered a trivial task. However, sometimes the information about word boundaries is not encoded in the surface form (as in Chinese script), or orthographic words are too coarse-grained and need to be further analyzed — which is the case for

many texts written in Ainu. In order to effectively process such texts, one needs to identify the implicit word boundaries.

The main contributions of this work are:

(i) fast n-gram model yielding results comparable to state-of-the-art systems in the task of word segmentation of the Ainu language;
(ii) open source implementation (https://github.com/karol-nowakowski/MiNgMatchSegmenter);
(iii) comparison of 4 different segmenters, including lexical n-gram models and a neural model performing word segmentation in the form of character sequence labelling.

The remainder of this paper is organized as follows. In Section 2 we describe the problem of word segmentation in the Ainu language. In Section 3 we review the related work. Section 4 explains the proposed approach to word segmentation. In Section 5 we introduce the Ainu language resources used in this research. This section also provides a description of word segmentation models applied in our experiments, as well as evaluation metrics. In Section 6 we analyze the experimental results. Finally, Section 7 contains conclusions and ideas for future improvements.

2. Word Segmentation in the Ainu Language

Ainu is an agglutinative language exhibiting some of the characteristics associated with polysynthesis, such as pronominal marking and noun incorporation (especially in the language of classical Ainu literature [1] (p. 5)). The following example demonstrates noun incorporation in Ainu:

kotan	apapa	ta	a=eponciseanu [2] (p. 111)
kotan	apa-pa	ta	a-e-pon-cise-anu
village	entrance-mouth	at	we/people-for[someone]-small-house-lay

"We built a small hut for [her] at the entrance to the village."

Ainu verbs and nouns combine with a variety of affixes—marking reciprocity, causativity, plurality and other categories—as well as function words: adnouns, verb auxiliaries and various types of postpositions, among others (in her analysis of the Shizunai dialect of Ainu, Kirsten Refsing [3] refers to both groups of grammatical morphemes with a common term: "clitics").

Most written documents in the Ainu language are transcribed using Latin alphabet, Japanese *katakana* script, or a combination of both (all textual data used in this research is written in Latin script). After two centuries of constant evolution—with multiple alternative notation methods being simultaneously in use—Ainu orthography has been, to a certain degree, standardized. One of the milestones in that process was a textbook compiled by the Hokkaido Utari Association (now Hokkaido Ainu Association) in cooperation with Ainu language scholars, published in 1994 under the title *Akor itak* ("our language") [4]. It was intended for the use in Ainu language classes held throughout Hokkaido and included a set of orthographic rules for both Latin alphabet and *katakana*-based transcription. They are widely followed to this day, for example by Hiroshi Nakagawa [5], Suzuko Tamura [2] and the authors of the Topical Dictionary of Conversational Ainu [6]. For detailed analyses of notation methods employed by different authors and how they changed with time, please refer to Kirikae [7], Nakagawa [8] and Endō [9].

Concerning word segmentation, however, no standard guidelines have been established to date [10] (p. 198), [11] (p. 5), and polysynthetic verb morphology only adds to the confusion [5] (p. 5). Contemporary Ainu language experts—while taking different approaches to handling certain forms, such as compound nouns and lexicalized expressions—generally treat morphemes entering in syntactic relations with other words as distinct units, even if they are cliticized to a host word in the phonological realization. In dictionaries, study materials and written transcripts of Ainu oral tradition that were published in the last few decades [2,12,13], it is a popular practice to use *katakana* to reflect pronunciation, while parallel text in Latin characters represents the underlying forms. The problem

is most noticeable in older documents and texts written by native speakers without a background in linguistics, who tended to divide text into phonological words or larger prosodic units (sometimes whole verses)—see Sunasawa [10] (p. 196). As a consequence, orthographic words in their notation comprise, on average, more morphemes. This, in turn, leads to an increase in the proportion of items not to be found in the existing dictionaries, which makes the content of such texts difficult to comprehend by less experienced learners. Furthermore, in the context of Natural Language Processing it renders the already limited data even more sparse. In order to facilitate the analysis and processing of such documents, a mechanism for word boundary detection is necessary.

3. Related Work

Existing approaches to the problem of tokenization and word segmentation can be largely divided into rule-based and data-driven methods. Data-driven systems may be further subdivided into lexicon-based systems and those employing statistical language models or machine learning.

In space-delimited languages, rule-based tokenizers—such as the Stanford Tokenizer (https://nlp.stanford.edu/software/tokenizer.html; accessed on 26 September 2019) [14]—are sufficient for most applications. On the other hand, in languages where word boundaries are not explicitly marked in text (such as Chinese and Japanese), word segmentation is a challenging task, receiving a great deal of attention from the research community. For such languages, a variety of data-driven word segmentation systems have been proposed. Among dictionary-based algorithms, one of the most popular approaches is the longest match method (also referred to as the maximum matching algorithm or MaxMatch) [15] and its variations [16,17]. In more recent work, however, statistical and machine learning methods prevail [18–22]. Furthermore, as in many other Natural Language Processing tasks, the past few years have witnessed an increasing interest towards artificial neural networks among the researchers studying word segmentation, especially for Chinese. A substantial part of the advancements in this area stem from using large external resources, such as raw text corpora, for pretraining neural models [23–27]. Unfortunately, such large-scale data is not available for many lesser-studied languages, including Ainu. For Japanese and Chinese, word segmentation is sometimes modelled jointly with part-of-speech tagging, as the output of the latter task can provide useful information to the segmenter [21,28–30].

Outside of the East Asian context, word segmentation-related research is focused mainly on languages with complex morphology and/or extensive compounding—such as Finnish, Turkish, German, Arabic and Hebrew—where splitting coarse-grained surface forms into smaller units leads to a significant reduction in the vocabulary size and thus lower proportion of out-of-vocabulary words [31–35]. Apart from that, even in languages normally using explicit word delimiters, there exist special types of text specific to the web domain, such as Uniform Resource Locators (URL) and *hashtags*, whose analysis requires the application of a word segmentation procedure [35,36].

In 2016 Grant Jenks released WordSegment—a Python module for word segmentation, utilizing a Stupid Backoff model (http://www.grantjenks.com/docs/wordsegment/; accessed on 26 September 2019). Due to relatively low computational cost, Stupid Backoff [37] is good for working with extremely large models, such as the Google's trillion-word corpus (https://ai.googleblog.com/2006/08/all-our-n-gram-are-belong-to-you.html; accessed on 26 September 2019) used as WordSegment's default training data. In terms of the model's accuracy, however, other language modelling methods—in particular the approach proposed by Kneser and Ney [38] and enhanced by Chen and Goodman [39]—proved to perform better, especially with smaller amounts of data [37]. For that reason, in this research, apart from comparing our word segmentation algorithm to WordSegment, we carried out additional experiments with a segmentation algorithm based on an n-gram model with modified Kneser-Ney smoothing. In the context of word segmentation, Kneser-Ney smoothing has previously been used by Doval and Gómez-Rodríguez [35].

Apart from models concerned directly with words, a widely practised approach to word segmentation is to define it as a character sequence labelling task, where each character is assigned with

a tag representing its position in relation to word boundaries. While the early works belonging to this category relied on "traditional" classification techniques, such as maximum entropy models [40] and Conditional Random Fields [41], in recent studies neural architectures are being actively explored [23,27,28,30,42]. In 2018, Shao et al. [43] released a language-independent character sequence tagging model based on recurrent neural networks with Conditional Random Fields interface, designed for performing word segmentation in the Universal Dependencies framework. It obtained state-of-the-art accuracies on a wide range of languages. One of the key components of their methodology (originally proposed in [30]) are the concatenated n-gram character representations, which offer a significant performance boost in comparison to conventional character embeddings, without resorting to external data sources. We used their implementation in the experiments described later in this paper, in order to verify how a character-based neural model performs under extremely low-resource conditions, such as those of the Ainu language, and how it compares with segmenters utilizing lexical n-grams, including ours.

To address the problem of word segmentation in the Ainu language, Ptaszynski and Momouchi [44] proposed a segmenter based on the longest match method. Later, Ptaszynski et al. [45] investigated the possibility of improving its performance by expanding the dictionary base used in the process. Nowakowski et al. [46] developed a lexicon-based segmentation algorithm maximizing mean token length. Finally, Nowakowski et al. [47] proposed a segmenter searching for the minimal sequence of n-grams matching the input string, an early and less efficient version of the MiNgMatch algorithm presented in this paper.

4. Description of the Proposed Approach

In the proposed method, we reduce the problem of word segmentation to that of finding the shortest sequence of lexical n-grams matching the input string. For each space-delimited segment in the input text, our algorithm finds a single n-gram or the shortest sequence of n-grams, such that after concatenation and removing all whitespaces is equal to that input segment. In cases where multiple segmentation paths with the same number of n-grams are possible, the sequence with the highest score is selected. Scoring function, given a candidate sequence S, can be defined as below:

$$Score(S) = \prod_{s \in S} \frac{Count(s)}{N} \tag{1}$$

where $Count(\cdot)$ denotes the frequency of a particular n-gram in the training corpus and N is the total number of n-grams in that corpus not exceeding the maximum n-gram order specified for the model.

If the model is unable to match any n-gram to the given string or its part, it is treated as an out-of-vocabulary item and returned without modification. Furthermore, the user may specify the maximum number of n-grams to be used in the segmentation of a single input segment. Strings for which the algorithm could not match a sequence of n-grams equal to or shorter than the limit, are retained without modification. The only exception to that rule are punctuation marks—they are separated from alpha-numeric strings in a post-processing step.

4.1. N-gram Data

Listing 1 shows a sample from the data used by our model. The first column contains unsegmented strings used by the matching algorithm, each of them corresponding to a lexical n-gram. In the rightmost column, we store precomputed scores, represented as logarithms.

Once the best sequence of n-grams has been selected, the indices of word boundaries for each n-gram (stored in the second column) are used to produce the final segmentation. For cases where multiple n-gram patterns recorded in the training corpus resulted in the same string after removing whitespaces from between the tokens, we only included the most frequent segmentation in the model. For instance, the 3-gram *aynu mosir ka* ("the land of Ainu also")—which appeared in the data

6 times—was pruned, as the bigram variant *aynumosir ka* was more frequent, with 63 occurrences. An alternative segmentation can still be returned by the segmenter if it is more frequent in a longer context. For example, although the preferred segmentation of the string *ciki* is *ciki* ("if"; 594 instances), rather than *ci ki* (first person pronominal marker *ci* attached to auxiliary verb *ki*) with 32 occurrences, in the case of a longer segment: *cikisiri*, the only segmentation attested in our data is *ci ki siri* (*ci ki* followed by the nominalizing evidential particle *siri*), appearing 3 times in the training corpus.

Listing 1: Sample from the n-gram data used by MiNgMatch Segmenter.

```
N-gram (unsegmented):    Indices of word    Score:
boundaries:
koranan                  5,3                -3.34220648264453
penekusu                 4,2                -3.3427874786936
korwa                    3                  -3.3439518077607
patek                                       -3.34628987252967
```

4.2. Computational Cost

The maximum number of candidate segmentations to be generated by our algorithm, given a string composed of n characters, can be calculated as follows:

$$\sum_{k=0}^{min\{m,l\}} \binom{n-1}{k} \tag{2}$$

where m stands for the smallest number of n-grams existing in the model needed to create a sequence matching the input string, and l represents the limit of n-grams per input string (specified by the user), such that $l \leq n$. In practice it means that, apart from rare situations where only a sequence of single-character unigrams can be matched to the given string, our algorithm has a lower computational cost than a model which considers all the 2^{n-1} possible segmentations (obviously, a word segmentation algorithm evaluating each unique segmentation path would be highly impractical; a typical approach, also taken by us, is to reduce that number by applying dynamic programming and memoization techniques).

5. Data and Experiments

5.1. Training Data

Language models applied in this research were trained on Ainu language textual data from eight different sources:

(A) *Ainu Shin'yōshu* [48] (SYOStrain)

A collection of 13 mythic epics (*kamuy yukar*) compiled by Yukie Chiri. In the training of our models we used a version with modernized transcription, published by Hideo Kirikae [49]. We only included 11 epics in the training set, while the remaining 2 texts were used as test data (see the next section).

(B) A Talking Dictionary of Ainu: A New Version of Kanazawa's Ainu Conversational dictionary [50] (TDOA)

An online dictionary based on the *Ainugo kaiwa jiten* [51], a dictionary compiled by Shōzaburō Kanazawa and Kotora Jinbō, and published in 1898. It contains 3,847 entries, each of them consisting of a single word, multiple related words, a phrase or a sentence. For training we used the modernized transcription produced by Bugaeva et al. [50]. The last 285 entries (roughly 10% of the dictionary, character-wise) were excluded from the training data, in order to use them as test data in evaluation experiments (see the next section).

(C) Glossed Audio Corpus of Ainu Folklore [52] (GACF)

A digital collection of 10 Ainu folktales with glosses (morphological annotation) and translations into Japanese and English.

(D) Dictionary of Ainu place names [53] (MOPL)

A dictionary of Ainu place names in the form of a database. It includes a total of 3,152 topological names, along with the analysis of their components and Japanese translations.

(E) Dictionary of the Mukawa dialect of Ainu [54] (MUKA)

An online resource developed on the basis of 150 hours of speech in the Ainu language (Mukawa dialect), recorded by Tatsumine Katayama between 1996 and 2002 with two native speakers: Seino Araida and Fuyuko Yoshimura. It contains 6,284 entries.

(F) Collection of Ainu Oral Literature [55] (NIBU)

A digital collection of 98 texts in Ainu: 53 prose tales (*uwepeker*), 29 mythic epics (*kamuy yukar*), 11 heroic epics (*yukar*) and 5 texts of other types. For detailed description (in Japanese) see: http://www.town.biratori.hokkaido.jp/biratori/nibutani/pdf/ainu_gaiyou.pdf (accessed on 26 September 2019).

(G) Ainu Language Archive – Materials [56] (AASI)

An online archive of texts in Ainu, based upon voice and video recordings taken from three native speakers: Matsuko Kawakami, Toshi Ueda and Tatsujirō Kuzuno.

(H) Ainu Language Archive – Dictionary [56] (AAJIhw)

An online lexicon consisting of digitized versions of three Ainu-Japanese dictionaries [2,57,58], comprising a total of 33,126 entries. We used only the headwords included in the dictionary. After removing duplicates (homographic entries), a total of 16,107 entries remained.

The following post-processing steps were applied to the training corpus:

(1) The data was cleaned, resulting in files containing only raw Ainu text in Latin alphabet.
(2) Accented vowels (⟨á⟩, ⟨é⟩, ⟨í⟩, ⟨ó⟩, ⟨ú⟩) used in some materials were replaced with their unaccented counterparts (⟨a⟩, ⟨e⟩, ⟨i⟩, ⟨o⟩, ⟨u⟩).
(3) Underscores (⟨_⟩) used in some materials to indicate phonological alternations were removed.
(4) Equality signs (⟨=⟩) used to denote personal markers were either removed, or replaced with whitespaces (if there was no whitespace in the original text). While their presence is an unambiguous indicator of a boundary between two tokens (although they are traditionally referred to as "affixes", we treat those morphemes as separate units, which is a common practice among present-day experts; for a detailed analysis of their morphological status, please refer to Bugaeva [59]), they were not used in older texts, which are going be the main target of a word segmentation system, therefore we decided to exclude them from the data. This resulted in a corpus of text comprising a total of 481,291 segments (space-delimited units of text). The statistics of all eight datasets after this step are shown in Table 1.
(5) Finally, punctuation marks were separated from words. However, non-alphanumeric characters used word-internally (e.g., hyphens indicating boundaries between the constituents of compound words and apostrophes representing glottal stop) were not modified.

Table 1. Statistics of Ainu text collections and dictionaries used as the training data.

Text Collection:	SYOStrain	TDOA	GACF	MOPL	MUKA	NIBU	AASI	AAJIhw	Overall
Characters (excluding spaces):	36,780	54,545	82,436	26,872	317,099	459,253	803,945	131,449	1,912,379
Segments:	8786	12,978	22,559	9246	71,232	122,314	218,069	16,107	481,291
Avg. segment length:	4.186	4.203	3.654	2.906	4.452	3.755	3.687	8.161	3.973

5.2. Test Data

Two different sets of held-out data were used in our evaluation experiments:

(A) *Ainu Shin'yōshu* [48] (**SYOS**)

This dataset consists of the two yukar epics from *Ainu Shin'yōshu* which were excluded from the training data:

- "Esaman yaieyukar, 'Kappa reureu kappa' " ("'Kappa reureu kappa', the song sung by the otter");
- "Pipa yaieyukar, 'Tonupeka ranran' " ("'Tonupeka ranran', the song sung by the marsh shellfish").

(B) *Ainugo Kaiwa Jiten* [51] (**AKJ**)

The portion of the original dictionary corresponding to the entries from A Talking Dictionary of Ainu which we removed from the training data, was applied as the second evaluation dataset.

In the test data we retained the word segmentation of the original transcriptions (by Chiri [48] and Jinbō and Kanazawa [51]). However, in order to prevent differences in spelling from affecting the word segmentation algorithm's performance, the text was preprocessed by unifying its spelling with modern versions transcribed by Kirikae [49] and Bugaeva et al. [50] (the task of spelling modernization is out of the scope of this paper and will be addressed separately.). In the case of the *Ainugo kaiwa jiten* and TDOA, there were also some differences in the usage of punctuation marks as well as several words which appeared in the original text, but the authors of the modernized transcription decided to remove them – in such cases the text was unified with the modern transcription, with the exception of equality signs attached to personal markers, which were omitted. A sample sentence from the *Ainugo kaiwa jiten* before and after this preprocessing step is shown in Table 2. Table 3 presents the statistics of both evaluation datasets, in comparison with the portions of modernized texts corresponding to them.

Table 2. A fragment from the Talking Dictionary of Ainu (TDOA)/Ainugo kaiwa jiten (AKJ) dataset.

***Ainugo kaiwa jiten* [51]:**	Nepka ayep an shiri he an?
TDOA [50]:	nep ka a= ye p an siri an?
AKJ:	nepka ayep an siri an?
Translation [50]:	Is there something you (wanted) to say?

Table 3. Statistics of the samples used for evaluation and their modern transcription equivalents.

Data	Characters (Excluding Spaces)	Segments	Avg. Segment Length	OoV* Rate
SYOS:	2667	441	6.070	0.259
Kirikae [49]:		643	4.163	0.034
AKJ:	4840	1143	4.234	0.044
Bugaeva et al. [50]:		1259	3.844	0.005

* **OoV**—out-of-vocabulary words.

5.3. Experiment Setup

In our experiments we tested the following word segmentation systems:

(1) a corpus-based word segmentation algorithm minimizing the number of n-grams needed to match the input string (MiNgMatch Segmenter);
(2) a segmentation algorithm with Stupid Backoff language model (WordSegment with modifications);

(3) a segmentation algorithm with a language model applying modified Kneser-Ney smoothing (later referred to as "mKN");
(4) a segmentation system based on character sequence labelling using a neural model [43] (later we will refer to it as "Universal Segmenter").

5.4. MiNgMatch Segmenter

Our algorithm was tested in two variants:

- with the limit of n-grams per input segment equal to the number of characters in the input string;
- with the limit of n-grams per input segment set to 2 (based on the observation that in most cases where a single input segment is divided into 3 or more n-grams, that segmentation is incorrect).

In experiments conducted by Nowakowski et al. [47] with an early version of the segmenter, the best results were in most cases yielded with the order of n-grams not exceeding 5-grams. Thus, for the n-gram models examined in the present paper we set the limit of n to 5.

5.5. WordSegment (Stupid Backoff Model)

WordSegment is an open source Python module for word segmentation developed by Grant Jenks, based on the work of Peter Norvig [60]. In our evaluation experiments we applied the system with two modifications:

(A) We added the option of using n-gram models with the order of n-grams higher than 2 (in original WordSegment, only unigrams and bigrams were used, whereas we wanted to test models with the order of up to 5).
(B) We added the possibility of manipulating the backoff factor. Although it was a part of the original formulation by Brants et al. [37], Peter Norvig and Grant Jenks omitted it from their implementations.

We examined three different values of the backoff factor:

- 1 (i.e. no backoff factor, as in original WordSegment);
- 0.4 (as suggested by Brants et al. [37]; later we will refer to this model as "SB-0.4");
- 0.09, only applied to 1-grams (this configuration achieved the best F-score in our preliminary experiments, at the cost of lower Precision). Let w_i denote a candidate word to be evaluated in the context of k previous words (w_{i-k}^{i-1}). The recursive scoring function employed in this variant (later referred to as "SB-0.09") can be defined as follows:

$$Score(w_i, k) = \begin{cases} \frac{Count(w_{i-k}^{i})}{Count(w_{i-k}^{i-1})} & \text{if } Count(w_{i-k}^{i}) > 0 \\ \begin{cases} Score(w_i, k-1) & \text{if } k \geq 2 \\ \alpha \frac{Count(w_i)}{N_1} & \text{otherwise.} \end{cases} & \text{otherwise.} \end{cases} \quad (3)$$

with α representing the backoff factor specified for unigrams, and N_1 being the total number of unigrams in the training corpus.

5.6. Segmenter with Language Model Applying Modified Kneser-Ney Smoothing

In the next experiment, we tested a word segmentation system similar to WordSegment (the same dynamic programming algorithm is used to generate candidate segmentations), but employing a language model with modified Kneser-Ney smoothing for choosing the most probable segmentation path. The model was generated using the KenLM Language Model Toolkit (https://kheafield.com/code/kenlm/; accessed on 26 September 2019).

Analogically to the experiments with our system and WordSegment, we used language models with the maximum order of n-grams set to 5.

5.7. Universal Segmenter

Apart from segmenters utilizing language models based on lexical n-grams, we carried out a series of experiments using the character-level sequence labelling model developed by Yan Shao et al. [43]. We trained the model in three different variants:

5.7.1. Default Model (henceforth, "US-Default")

In this experiment, we applied the default training settings, designed to work with space-delimited languages. The same training data as in previous experiments was used, which means that tokens in the training set correspond to those in gold standard data.

5.7.2. With Spaces Ignored (henceforth, "US-ISP")

Here, we trained the model with the `-isp` argument. It results in the removal of space delimiters from the training data, which means it is effectively treated in a similar way to Chinese or Japanese script.

5.7.3. With Multi-Word Tokens (later referred to as "US-MWTs_rnd" and "US-MWTs")

When processing older Ainu texts, many space-delimited segments need to be split in multiple tokens. Consequently, the default model relying on whitespaces and trained on the data with modern segmentation is ineffective. Unlike in Chinese or Japanese, however, a large portion of word boundaries is already correctly indicated by whitespaces in the input text, so ignoring them altogether (as in US-ISP models) is not the optimal method, as well. In order to create a model better suited to our task, we used the concept of multi-word tokens (https://universaldependencies.org/u/overview/tokenization.html; accessed on 26 September 2019) existing in Universal Dependencies and also reflected in the Universal Segmenter.

Firstly, we converted the two datasets (SYOStrain and TDOA) for which both old and modernized transcriptions exist, to a format where boundaries between words grouped together as a single space-delimited string in the original transcription are treated as boundaries between components of a multi-word token. For the remaining six datasets, however, only a single transcription by contemporary experts is available. We therefore applied the following two methods to simulate sparser word segmentation of old texts by generating multi-word tokens artificially:

(A) As a baseline method, we created multi-word tokens in a random manner. Namely, we assigned each whitespace in the data with a 50% chance of being removed and thus becoming a boundary between components of a multi-word token. This resulted in the generation of 105,663 multi-word tokens. Later we will refer to the models learned from this version of the training data as "US-MWTs_rnd".

(B) In the second approach, multi-word tokens were generated in a semi-supervised manner using the Universal Segmenter itself. To achieve that, we converted multi-word tokens previously identified in SYOStrain and TDOA to multi-token words (defined in the UD scheme as words consisting of multiple tokens, but treated as a single syntactic unit) and trained a word segmentation model on these two datasets. The resulting model was then used to process the rest of the training corpus. As a result, some tokens were grouped in multi-token words (a total of 70,373 such words were generated). In the final step, we converted the multi-token words to multi-word tokens. This variant of the data was used to train the group of models later referred to as "US-MWTs".

We illustrate the operations described above in Table 4, using a sample from the SYOStrain dataset.

Table 4. Operations on training data for the Universal Segmenter.

Original text [48]:
Shineanto ta shirpirka kusu [...] (“One day, since the weather was nice [...]”)
Modernized transcription [49]:
Sine an to ta sir pirka kusu [...]
Training data (in CoNLL-U format *) with multi-word tokens:

`1-3 sineanto _ _ _ _ _ _ _ _ _`
`1 sine _ _ _ _ _ _ _ _ _`
`2 an _ _ _ _ _ _ _ _ _`
`3 to _ _ _ _ _ _ _ _ _`
`4 ta _ _ _ _ _ _ _ _ _`
`5-6 sirpirka _ _ _ _ _ _ _ _ _`
`5 sir _ _ _ _ _ _ _ _ _`
`6 pirka _ _ _ _ _ _ _ _ _`
`7 kusu _ _ _ _ _ _ _ _ _`

Training data (in CoNLL-U format *) with multi-token words:

`1 sine an to _ _ _ _ _ _ _ _ _`
`2 ta _ _ _ _ _ _ _ _ _`
`3 sir pirka _ _ _ _ _ _ _ _ _`
`4 kusu _ _ _ _ _ _ _ _ _`

* See: https://universaldependencies.org/format.html (accessed on 26 September 2019).

Apart from simple character embeddings, the Universal Segmenter allows the usage of concatenated n-gram vectors encoding rich local information. We investigated the performance with 3-, 5-, 7-, 9- and 11-grams. Any parameters of the training process not mentioned above were set to default values.

5.8. Evaluation Method

In order to evaluate word segmentation performance, we employed three metrics: Precision (P), Recall (R) and balanced F-score (F_1). Precision is defined as the proportion of correct word boundaries (whitespaces) within all word boundaries returned by the system (B_s), whereas Recall is the portion of word boundaries present in expert-annotated data (B_e) which were also correctly predicted by the segmenter. The balanced F-score is the harmonic mean of Precision and Recall.

$$P = \frac{|B_s \cap B_e|}{|B_s|} \tag{4}$$

$$R = \frac{|B_s \cap B_e|}{|B_e|} \tag{5}$$

$$F_1 = 2\frac{PR}{P+R} \tag{6}$$

In addition, we evaluated word-level Accuracy for OoV words, defined as the proportion of unseen tokens in expert-annotated data (U_e), correctly segmented by the system (U_s):

$$Accuracy = \frac{|U_s|}{|U_e|} \tag{7}$$

6. Results and Discussion

The results of the evaluation experiments with our algorithm are presented in Table 5. The variant without the limit of n-grams per input segment produces unbalanced results (especially on SYOS), with relatively low Precision. After setting the limit to 2, Precision improves at the cost of a drop in Recall. The F-score is better for SYOS, while on AKJ there is a very slight drop.

Table 6 shows the results of experiments with the Stupid Backoff model. When no backoff factor is applied, results for both test sets are similar to those from the MiNgMatch Segmenter without the limit of n-grams per input segment. Setting the backoff factor to an appropriate value allows for significant improvement in Precision and F-score (and in some cases also small improvements in Recall). For the F-score, it is better to set a low backoff factor (e.g., 0.09) for 1-grams only, than to set it to a fixed value for all backoff steps (e.g., 0.4, as Brants et al. [37] did). A backoff factor of 0.4 gives significant improvement in Precision with higher order n-gram models, but at the same time Recall drops drastically and overall performance deteriorates. For models with an n-gram order of 3 or higher, the backoff factor has a bigger impact on the results than further increasing the order of n-grams included in the model. A comparison with the results yielded by MiNgMatch shows that setting the limit of n-grams per input segment is more effective than Stupid Backoff as a method for improving precision of the segmentation process—it leads to a much smaller drop in Recall.

The results of the experiment with models employing modified Kneser-Ney smoothing are shown in Table 7. They achieve higher Precision than both the other types of n-gram models. Nevertheless, due to very low Recall, the overall results are low.

The results obtained by the Universal Segmenter are presented in Table 8. The default model (regardless of what kind of character representations are used—conventional character embeddings or concatenated n-gram vectors) learns from the training data that the first and the last character of a word (corresponding to `B`, `E` and `S` tags) are always adjacent either to the boundary of a space-delimited segment or to a punctuation mark. As a result, the model separates punctuation from alpha-numeric strings found in the input, but never applies further segmentation to them.

Table 5. Evaluation results—MiNgMatch Segmenter (best results in bold).

Max. N-gram Order:			**2**		**3**		**4**		**5**	
Test Data:			**SYOS**	**AKJ**	**SYOS**	**AKJ**	**SYOS**	**AKJ**	**SYOS**	**AKJ**
Max. n-grams per segment	**Max.** *	**Precision**	0.918	0.968	0.923	0.969	0.925	0.969	0.923	0.969
		Recall	0.918	0.986	0.943	0.989	0.952	0.989	0.952	0.990
		F-score	0.918	0.977	0.933	0.979	0.938	0.979	0.938	**0.980**
	2	**Precision**	0.952	0.972	0.955	0.972	0.957	0.972	0.956	0.973
		Recall	0.899	0.981	0.924	0.985	0.933	0.985	0.933	0.986
		F-score	0.925	0.976	0.939	0.978	**0.945**	0.979	0.944	0.979

* **Max.**—number of characters in the input segment.

Table 6. Evaluation results—Stupid Backoff model (best results in bold).

Max. N-gram Order:			**2**		**3**		**4**		**5**	
Test Data:			**SYOS**	**AKJ**	**SYOS**	**AKJ**	**SYOS**	**AKJ**	**SYOS**	**AKJ**
Backoff factor	**1**	**Precision**	0.915	0.965	0.923	0.965	0.923	0.965	0.923	0.965
		Recall	0.916	0.986	0.958	0.989	0.961	0.988	0.961	0.988
		F-score	0.915	0.975	0.940	**0.977**	0.942	0.976	0.942	0.976
	0.4	**Precision**	0.924	0.967	0.934	0.968	0.952	0.972	0.958	0.974
		Recall	0.921	0.988	0.944	0.986	0.913	0.979	0.875	0.966
		F-score	0.922	**0.977**	0.939	**0.977**	0.932	0.975	0.915	0.970
	$n = 1$: **0.09** $n > 1$: **1**	**Precision**	0.934	0.968	0.937	0.968	0.937	0.965	0.937	0.965
		Recall	0.927	0.986	0.961	0.987	0.963	0.986	0.963	0.986
		F-score	0.930	**0.977**	0.949	**0.977**	**0.950**	0.975	**0.950**	0.975

Table 7. Evaluation results—model with Kneser-Ney smoothing (best results in bold).

Max. N-gram Order:	**2**		**3**		**4**		**5**	
Test Data:	**SYOS**	**AKJ**	**SYOS**	**AKJ**	**SYOS**	**AKJ**	**SYOS**	**AKJ**
Precision	0.970	0.979	0.975	0.978	0.975	0.979	0.975	0.979
Recall	0.693	0.946	0.724	0.944	0.726	0.943	0.726	0.943
F-score	0.808	**0.962**	0.831	0.961	**0.832**	0.961	**0.832**	0.961

Table 8. Evaluation results—Universal Segmenter (best results in bold).

Model Version:			**Default**		**ISP**		**MWTs_rnd**		**MWTs**	
Test Data:			**SYOS**	**AKJ**	**SYOS**	**AKJ**	**SYOS**	**AKJ**	**SYOS**	**AKJ**
Order of concatenated n-gram vectors	**–**	**F-score**	0.809	0.931	0.874	0.950	-	-	-	-
	3	**F-score**	0.813	0.931	0.894	0.946	0.939	0.971	0.938	0.974
	5	**F-score**	0.812	0.931	0.888	0.958	0.945	0.973	0.948	0.978
	7	**F-score**	0.812	0.931	0.897	0.955	0.944	0.973	0.948	0.977
	9	**Precision**	0.998	0.981	0.910	0.952	0.950	0.972	0.976	0.980
		Recall	0.686	0.885	0.910	0.962	0.947	0.976	0.927	0.979
		F-score	0.813	0.931	0.910	0.957	0.948	0.974	**0.951**	**0.979**
	11	**F-score**	0.812	0.931	0.914	0.954	0.946	0.974	0.950	0.975

US-ISP models are better but still notably worse than lexical n-gram models (especially on SYOS). Unlike with default settings, the model trained on data without whitespaces learns to predict word boundaries within strings of alpha-numeric characters. However, when presented with test data including spaces, they impede the segmentation process rather than supporting it. As shown in Table 9, if we only take into account the word boundaries not already indicated in the raw test set, the model makes more correct predictions in data where the whitespaces have all been removed.

Table 9. US-ISP model (with 9-gram vectors): F-score for word boundaries not indicated in original transcription.

Spaces in Test Data Retained	**Test Data**	
	SYOS	**AKJ**
YES	0.811	0.880
NO	0.844	0.930

Models with multi-word tokens achieve significantly higher results. Precision of the US-MWTs model is on par with the segmenter applying Kneser-Ney smoothing, while maintaining relatively high Recall. It yields lower Recall than the model with randomly generated multi-word tokens, but the F-score is higher due to better Precision.

With the exception of the US-ISP model on SYOS, all variants of the neural segmenter achieved the best performance with concatenated 9-gram vectors. This contrasts with the results reported by Shao et al. [30] for Chinese, where in most cases there was no further improvement beyond 3-grams. This behavior is a consequence of differences between writing systems: words in Chinese are on average composed of less characters than in languages using alphabetic scripts. Due to a much bigger character set size, *hanzi* characters are also more informative to word segmentation [43], hence better performance with models using shorter context.

6.1. General Observations

Due to data sparsity, n-gram coverage in the test set (the fraction of n-grams in the test data that can be found in the training set) is low (see Table 10). It means that many multi-word tokens from the test set are known to n-gram models as separate unigrams, but not in the form of a single

n-gram. The Stupid Backoff model with a backoff factor for unigrams set to a moderate value (such as 0.09) is able to segment such strings correctly. However, it also erroneously segments some OoV single-word tokens whose surface forms happen to be interpretable as a sequence of concatenated in-vocabulary unigrams, resulting in lower Precision. On the other hand, models assigning low scores to unigrams (such as a 4- or 5-gram model with the Stupid Backoff and backoff factor set as suggested by Brants et al. [37], and in particular the model applying modified Kneser-Ney smoothing) are better at handling OoV words (see Table 11), but as a result of probability multiplication, in many cases they score unseen multi-word segments higher than a sequence of unigrams into which the given segment should be divided, hence yielding lower Recall.

Table 10. N-gram coverage.

N-gram Order:	**1**	**2**	**3**	**4**	**5**
Test data:	**N-gram coverage:**				
SYOS	0.966	0.627	0.338	0.188	0.128
AKJ	0.995	0.792	0.487	0.236	0.099

Table 11. Word-level Accuracy for OoV words (best models only).

Model:	**MiNgMatch**	**SB-0.4**	**SB-0.09**	**mKN**	**US-MWTs**
Test data:	**Accuracy:**				
SYOS	0.320	0.120	0.000	0.320	0.560
AKJ	0.000	0.000	0.000	0.125	0.625

Universal Segmenter operates at the level of characters rather than words, which makes it more robust against unseen words. This, along with the ability of neural models to transform discrete, sparse inputs into continuous representations capturing similarities between them, such as morphological features [35], explains the fact that it is able to achieve high Precision while maintaining relatively high Recall.

In line with these observations, we found Universal Segmenter to be the only segmenter in our experiments whose output includes tokens seen neither in the training data nor in the test set. For instance, it correctly segmented the input token *ekampaktehi* into *ekampakte hi* ("a promise"), whereas other systems either did not divide it at all, or segmented it into a sequence of in-vocabulary unigrams (e.g., *ekampak te hi*).

6.2. Error Comparison

Using the outputs of the best performing models, we measured how similar the errors made by different segmenters were. In particular, we calculated the Jaccard index between lists of errors found in each pair of outputs.

Results are presented in Table 12. Output of the model with modified Kneser-Ney smoothing is the least similar to most other models' outputs, which can be explained simply by the fact that it made the highest number of errors on both datasets (statistics are shown in Table 13). On the other hand, the Universal Segmenter's output, while containing numbers of errors comparable to those produced by the best performing n-gram models, also exhibits a low level of similarity to them.

Table 12. Error comparison (Jaccard index).

Test set: SYOS	**MiNgMatch**	**SB-0.09**	**mKN**
US-MWTs	0.220	0.153	0.195
mKN	0.176	0.159	
SB-0.09	0.505		
Test set: AKJ	**MiNgMatch**	**SB-0.09**	**mKN**
US-MWTs	0.414	0.390	0.474
mKN	0.369	0.355	
SB-0.09	0.714		

Table 13. Statistics of word segmentation errors.

Model:	**MiNgMatch**	**SB-0.09**	**mKN**	**US-MWTs**
Test data:		**Errors:**		
SYOS	71	66	189	62
AKJ	50	58	91	49

Indeed, qualitative analysis of segmentations generated by the neural model confirms that in some parts they are quite different from the predictions made by other models. For instance, the two segments *wenpuri enantuykasi* were correctly divided into *wen puri enan tuykasi* ("[her] face [took] the color of anger") only by the Universal Segmenter. All other models incorrectly split the word *tuykasi* (possessive form of the locative noun *tuyka*, meaning "on [the face]") into *tuyka si*, the reason being the fact that the n-gram *wen puri enan tuyka* is attested (with 4 instances) in the training set. Conversely, there are also some errors only made by the Universal Segmenter. For instance, it was the only system to divide the in-vocabulary word *ayapo* (exclamation of pain) into two tokens: *a* and *yapo*, out of which *yapo* does not appear in the training data. Another example is the phrase *ki aineno* ("eventually"), transcribed by Kirikae as *ki a ine no* (3 instances in the training set) and segmented in the same way by n-gram models, whereas the neural model treated the last two words as a single unit, *ineno*. This prediction, however, might be arguably considered correct, as there exists one instance of *ineno* in Kirikae's transcription, used in the same context (*iki a ineno*). Based on the observations described above, we believe that implementing an ensemble of an n-gram model and a character sequence labelling neural model shall be an interesting avenue for future work.

6.3. Results on SYOS with Two Gold Standard Transcriptions

As mentioned in Section 2, there is a certain amount of inconsistencies in word segmentation even between contemporary scholars of Ainu, which means they are also present in our data. With that in mind, we decided to cross-check the results of our experiments against an additional gold standard transcription. For that purpose we used an alternative modernized transcription of SYOS by Katayama [61].

Firstly, we compared Katayama's transcription with the version edited by Kirikae [49], using the same evaluation metrics as in previous experiments with segmentation algorithms. The results are presented in Table 14.

Table 14. Katayama's transcription evaluated against Kirikae's transcription.

Precision	**Recall**	**F-score**
0.979	0.941	0.960

Our assumption is that—in spite of making different decisions as to whether to group certain morphemes together or to treat them as separate units—both experts produced correct transcriptions. In order to investigate the effect of this phenomenon on our experiments, we re-evaluated the outputs of the best performing segmentation models using a combination of both experts' transcriptions as the gold standard data. This time, Precision was defined as the proportion of word boundaries predicted by the model that can be also found in either of the gold standard transcriptions:

$$P = \frac{|B_s \cap (B_{e_1} \cup B_{e_2})|}{|B_s|} \tag{8}$$

Analogically, Recall was defined as the proportion of word boundaries found in both variants of the gold standard which were also correctly predicted by the model:

$$R = \frac{|B_s \cap B_{e_1} \cap B_{e_2}|}{|B_{e_1} \cap B_{e_2}|} \tag{9}$$

Results are shown in Table 15. Apart from the model with Kneser-Ney smoothing, the results achieved by all models improved substantially. The highest gain was obtained for our algorithm—the result improved to such an extent that it ranked first in terms of F-score. A large share of that difference can be attributed to a single token, *awa* (a conjunction created by combining the perfective aspect marker *a* and a coordinative conjunctive particle *wa*), appearing a total of 17 times in the test set. The MiNgMatch algorithm, operating at the level of input segments, followed Katayama in not splitting *awa*, as it is more frequent in the training data, with 289 occurrences, than the 2-gram variant *a wa* (181 instances). Nevertheless, models considering a wider context preferred the latter option, which conforms with how Kirikae transcribed it.

Table 15. SYOS: outputs of the best models re-evaluated against combined gold standard data (Kirikae [49] + Katayama [61]). Best results in bold.

Model:	MiNgMatch	SB-0.4	SB-0.09	mKN	US-MWTs
Precision	0.965	0.969	0.949	**0.981**	0.980
Recall	0.954	0.880	**0.964**	0.718	0.933
F-score	**0.959**	0.922	0.956	0.829	0.956

6.4. Execution Speed

Table 16 compares the total time taken by each of the best performing models to process the two test sets. In the case of segmenters based on lexical n-grams, we used 5-gram models. The Universal Segmenter's speed was evaluated on the model trained with concatenated 9-gram vector representations. Experiments with n-gram models were carried out on a Windows machine with Intel Core i7 running at 1.90 GHz and 16 GB of RAM. The Universal Segmenter was tested on an Ubuntu machine with four GPUs (NVIDIA GeForce GTX 1080 Ti) and 128 GB of RAM. Each value represents an average of five consecutive runs. The results indicate that our algorithm is unrivalled in terms of speed.

Table 16. Execution times in seconds.

Model:	MiNgMatch	SB-0.09	mKN	US-MWTs
Time (s)	0.812	5.837	4.785	7.717

7. Conclusions and Future Work

In this paper, we introduced the MiNgMatch Segmenter: a data-driven word segmentation algorithm finding the minimal sequence of n-grams needed to match the input text. We compared our algorithm with segmenters utilizing two state-of-the-art n-gram language modelling techniques

(namely, the Stupid Backoff model and a model with modified Kneser-Ney smoothing), as well as a neural model performing word segmentation as character sequence labelling.

The evaluation experiments revealed that the proposed approach is capable of achieving overall results comparable with the other best-performing models, especially when we take into account the variance in notation of certain lexical items by different contemporary experts. Given its low computational cost and competitive results, we believe that MiNgMatch could be applied to other languages, and possibly to other Natural Language Processing problems, such as speech recognition.

In terms of precision of the segmentation process and accuracy for out-of-vocabulary words, the sequence labelling neural model turned out to be the best option. In order to achieve that, however, it needs to be presented with training data tailored to the task, closely mimicking the intended target data. To this end, we demonstrated that such data can be bootstrapped from a small amount of manually annotated text, using the Universal Segmenter itself.

Important tasks for the future include performing experiments with the proposed algorithm on other languages and implementing an ensemble segmenter combining an n-gram model (such as MiNgMatch) with a neural model performing word segmentation as character sequence labelling. Another area that requires improvement is the handling of OoV words. All lexical n-gram-based models applied in our experiments performed poorly in this aspect and our algorithm was not an exception. One possible way to increase the MiNgMatch Segmenter's robustness against unseen forms might be to utilize character n-grams instead of word n-grams.

Author Contributions: Conceptualization, M.P., F.M. and K.N.; methodology, M.P., F.M. and K.N.; software, K.N.; validation, F.M.; formal analysis, K.N.; investigation, K.N.; resources, F.M.; data curation, M.P. and K.N.; writing—original draft preparation, K.N.; writing—review and editing, M.P. and F.M.; visualization, K.N.; supervision, F.M. and M.P.; project administration, F.M. and M.P.

Funding: This research received no external funding.

Acknowledgments: The authors would like to thank the anonymous reviewers for their constructive feedback. We also thank Christopher Bozek for diligent proofreading of the manuscript. We are also grateful to Jagna Nieuważny and Ali Bakdur for useful discussions.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Shibatani, M. *The Languages of Japan*; Cambridge University Press: London, UK, 1990.
2. Tamura, S. *Ainugo Chitose Hōgen Jiten: The Ainu-Japanese Dictionary: Saru Dialect*; Sōfūkan: Tokyo, Japan, 1998.
3. Refsing, K. *The Ainu language. The Morphology and Syntax of the Shizunai Dialect*; Aarhus University Press: Aarhus, Denmark, 1986.
4. Hokkaidō Utari Kyōkai [Hokkaido Ainu Association]. *Akor Itak*; [our language]; Hokkaidō Utari Kyōkai: Sapporo, Japan, 1994.
5. Nakagawa, H. *Ainugo Chitose Hōgen Jiten*; [dictionary of the Chitose dialect of Ainu]; Sōfūkan: Tokyo, Japan, 1995.
6. National Institute for Japanese Language and Linguistics. A Topical Dictionary of Conversational Ainu. Available online: http://ainutopic.ninjal.ac.jp (accessed on 25 August 2017).
7. Kirikae, H. Ainu ni yoru Ainugo hyōki [transcription of the Ainu language by Ainu people]. *Koku-bungaku kaishaku to kanshō* **1997**, *62*, 99–107.
8. Nakagawa, H. Ainu-jin ni yoru Ainugo hyōki e no torikumi [efforts to transcribe the Ainu language by Ainu people]. In *Hyōki no Shūkan no Nai Gengo no Hyōki = Writing Unwritten Languages*; Shiohara, A.; Kodama, S., Eds.; Tōkyō gaikoku-go daigaku, Ajia/Afurika gengo bunka kenkyūjo: Tokyo, Japan, 2006; pp. 1–44.
9. Endō, S. Nabesawa Motozō ni yoru Ainugo no kana hyōki taikei: Kokuritsu Minzoku-gaku Hakubutsukan shozo hitsu-roku nōto kara [Ainu language notation method used by Motozo Nabesawa: from the written notes held by the National Museum of Ethnology]. *Kokuritsu Minzoku-gaku Hakubutsukan chōsa hōkoku* **2016**, *134*, 41–66.
10. Sunasawa, K. *Ku Sukup Oruspe*; [my life's story]; Miyama Shobō: Sapporo, Japan, 1983.
11. Satō, T. *Ainugo Bunpō no Kiso*; [basics of Ainu grammar]; Daigaku Shorin: Tokyo, Japan, 2008.

12. The Foundation for Ainu Culture. *Chūkyū Ainugo—Saru*; [intermediate Ainu course—Saru dialect]; The Foundation for Ainu Culture: Sapporo, Japan, 2014.
13. Sapporo Gakuin Daigaku, Jinbungaku-bu [Sapporo Gakuin University, Faculty of Humanities]. *Ainu Bunka ni Manabu*; [Learning from Ainu Culture]; Sapporo Gakuin Daigaku Seikatsu Kyōdō Kumiai: Ebetsu, Japan, 1990.
14. Manning, C.; Surdeanu, M.; Bauer, J.; Finkel, J.; Bethard, S.; McClosky, D. The Stanford CoreNLP Natural Language Processing Toolkit. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, Baltimore, MD, USA, 23–24 June 2014, pp. 55–60.
15. Wu, Z.; Tseng, G. Chinese Text Segmentation for Text Retrieval: Achievements and Problems. *J. Am. Soc. Inf. Sci.* **1993**, *44*, 532–542. [CrossRef]
16. Nagata, M. A Self-Organizing Japanese Word Segmenter using Heuristic Word Identification and Re-estimation. In Proceedings of the Fifth Workshop on Very Large Corpora, Beijing, China; Hong Kong, China, 18–20 August 1997.
17. Sassano, M. Deterministic Word Segmentation Using Maximum Matching with Fully Lexicalized Rules. In Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, Volume 2: Short Papers, Gothenburg, Sweden, 3–7 April 2014; Association for Computational Linguistics: Gothenburg, Sweden, 2014; pp. 79–83.
18. Papageorgiou, C.P. Japanese Word Segmentation by Hidden Markov Model. In Proceedings of the Workshop on Human Language Technology; Association for Computational Linguistics, HLT '94, Stroudsburg, PA, USA, 8–11 March 1994; pp. 283–288.
19. Palmer, D.D. A Trainable Rule-Based Algorithm for Word Segmentation. In Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics, Madrid, Spain, 11–12 July 1997; pp. 321–328.
20. Peng, F.; Feng, F.; McCallum, A. Chinese Segmentation and New Word Detection Using Conditional Random Fields. In Proceedings of the 20th International Conference on Computational Linguistics, COLING '04, Geneva, Switzerland, 23–27 August 2004; Association for Computational Linguistics: Stroudsburg, PA, USA, 2004.
21. Kudo, T.; Yamamoto, K.; Matsumoto, Y. Applying Conditional Random Fields to Japanese Morphological Analysis. In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, Barcelona, Spain, 25–26 July 2004; Association for Computational Linguistics: Barcelona, Spain, 2004; pp. 230–237.
22. Neubig, G.; Nakata, Y.; Mori, S. Pointwise Prediction for Robust, Adaptable Japanese Morphological Analysis. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, Human Language Technologies, Portland, OR, USA, 19–24 June 2011; Association for Computational Linguistics: Portland, OR, USA, 2011; pp. 529–533.
23. Pei, W.; Ge, T.; Chang, B. Max-Margin Tensor Neural Network for Chinese Word Segmentation. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Baltimore, MD, USA, 22–27 June 2014; Association for Computational Linguistics: Baltimore, MD, USA, 2014; pp. 293–303.
24. Cai, D.; Zhao, H. Neural Word Segmentation Learning for Chinese. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Berlin, Germany, 7–12 August 2016; pp. 409–420.
25. Yang, J.; Zhang, Y.; Dong, F. Neural Word Segmentation with Rich Pretraining. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Vancouver, BC, Canada, 30 July–4 August 2017; pp. 839–849.
26. Yang, J.; Zhang, Y.; Liang, S. Subword Encoding in Lattice LSTM for Chinese Word Segmentation. *arXiv* **2018**, arXiv:1810.12594.
27. Ma, J.; Ganchev, K.; Weiss, D. State-of-the-art Chinese Word Segmentation with Bi-LSTMs. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018; pp. 4902–4908.
28. Zheng, X.; Chen, H.; Xu, T. Deep Learning for Chinese Word Segmentation and POS Tagging. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, WA, USA, 18–21 October 2013; pp. 647–657.

29. Morita, H.; Kawahara, D.; Kurohashi, S. Morphological Analysis for Unsegmented Languages using Recurrent Neural Network Language Model. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–21 September 2015; pp. 2292–2297.
30. Shao, Y.; Hardmeier, C.; Tiedemann, J.; Nivre, J. Character-based Joint Segmentation and POS Tagging for Chinese using Bidirectional RNN-CRF. In Proceedings of the Eighth International Joint Conference on Natural Language Processing, Taipei, Taiwan, 27 November–1 December 2017.
31. Monroe, W.; Green, S.; Manning, C.D. Word Segmentation of Informal Arabic with Domain Adaptation. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Baltimore, MD, USA, 22–27 June 2014; Association for Computational Linguistics: Baltimore, MD, USA, 2014; pp. 206–211.
32. Ma, J.; Henrich, V.; Hinrichs, E. Letter Sequence Labeling for Compound Splitting. In Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology, Berlin, Germany, 11 August 2016; Association for Computational Linguistics: Berlin, Germany, 2016; pp. 76–81.
33. Björkelund, A.; Falenska, A.; Yu, X.; Kuhn, J. IMS at the CoNLL 2017 UD Shared Task: CRFs and Perceptrons Meet Neural Networks. In Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Vancouver, BC, Canada, 3–4 August 2017; Association for Computational Linguistics: Vancouver, BC, Canada, 2017; pp. 40–51.
34. Tuggener, D. *Proceedings of the 3rd Swiss Text Analytics Conference—SwissText 2018, CEUR Workshop Proceedings, Winterthur, Switzerland, 12–13 June 2018*; ZHAW Zurich University of Applied Sciences, Institute of Applied Information Technology (InIT): Winterthur, Switzerland, 2018; Volume 2226, pp. 42–49.
35. Doval, Y.; Gómez-Rodríguez, C. Comparing Neural- and N-Gram-Based Language Models for Word Segmentation. *J. Assoc. Inf. Sci. Technol.* **2019**, *70*, 187–197 [CrossRef] [PubMed]
36. Wang, K.; Thrasher, C.; Hsu, B.J.P. Web Scale NLP: A Case Study on Url Word Breaking. In Proceedings of the 20th International Conference on World Wide Web, WWW '11, Hyderabad, India, 28 March 28–1 April 2011; ACM: New York, NY, USA, 2011; pp. 357–366.
37. Brants, T.; Popat, A.C.; Xu, P.; Och, F.J.; Dean, J. Large Language Models in Machine Translation. In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), Prague, Czech Republic, 28–30 June 2007; Association for Computational Linguistics: Prague, Czech Republic, 2007; pp. 858–867.
38. Kneser, R.; Ney, H. Improved backing-off for M-gram language modeling. In Proceedings of the 1995 International Conference on Acoustics, Speech, and Signal Processing, Detroit, MI, USA, 9–12 May 1995; pp. 181–184.
39. Chen, S.F.; Goodman, J. An Empirical Study of Smoothing Techniques for Language Modeling. In Proceedings of the 34th Annual Meeting on Association for Computational Linguistics, ACL '96, Stroudsburg, PA, USA, 23–28 June 1996; Association for Computational Linguistics: Stroudsburg, PA, USA, 1996; pp. 310–318.
40. Xue, N. Chinese Word Segmentation as Character Tagging. In Proceedings of the Second SIGHAN Workshop on Chinese Language Processing-Volume 17 2003, Sapporo, Japan, 11–12 July 2003; pp. 29–48.
41. Zhao, H.; Huang, C.N.; Li, M.; Lu, B.L. Effective Tag Set Selection in Chinese Word Segmentation via Conditional Random Field Modeling. In Proceedings of the 20th Pacific Asia Conference on Language, Information and Computation, Wuhan, China, 1–3 November 2006; Huazhong Normal University: Wuhan, China, 2006; pp. 87–94.
42. Huang, W.; Cheng, X.; Chen, K.; Wang, T.; Chu, W. Toward Fast and Accurate Neural Chinese Word Segmentation with Multi-Criteria Learning. *arXiv* **2019**, arXiv:1903.04190.
43. Shao, Y.; Hardmeier, C.; Nivre, J. Universal Word Segmentation: Implementation and Interpretation. *Trans. Assoc. Comput. Linguist.* **2018**, *6*, 421–435 [CrossRef]
44. Ptaszynski, M.; Momouchi, Y. Part-of-Speech Tagger for Ainu Language Based on Higher Order Hidden Markov Model. *Expert Syst. Appl.* **2012**, *39*, 11576–11582. [CrossRef]
45. Ptaszynski, M.; Ito, Y.; Nowakowski, K.; Honma, H.; Nakajima, Y.; Masui, F. Combining Multiple Dictionaries to Improve Tokenization of Ainu Language. In Proceedings of the 31st Annual Conference of the Japanese Society for Artificial Intelligence, Tsukuba, Tokyo, 13–15 November 2017.

46. Nowakowski, K.; Ptaszynski, M.; Masui, F. Improving Tokenization, Transcription Normalization and Part-of-speech Tagging of Ainu Language through Merging Multiple Dictionaries. In Proceedings of 8th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics, Workshop on Language Technology for Less Resourced Languages (LT-LRL), Poznan, Poland, 17–19 November 2017; pp. 317–321.
47. Nowakowski, K.; Ptaszynski, M.; Masui, F. Word n-gram based tokenization for the Ainu language. In Proceedings of the International Workshop on Modern Science and Technology (IWMST 2018), Wuhan, China, 25–26 October 2018; pp. 58–69.
48. Chiri, Y. *Ainu Shin-yōshū;* [collection of Ainu mythic epics]; Kyōdo Kenkyūsha: Tokyo, Japan, 1923.
49. Kirikae, H. *Ainu Shin-yōshū Jiten: Tekisuto, Bumpō Kaisetsu Tsuki;* [lexicon to Yukie Chiri's *Ainu Shin-yōshū* with text and grammatical notes]; Daigaku Shorin: Tokyo, Japan, 2003.
50. Bugaeva, A.; Endō, S.; Kurokawa, S.; Nathan, D. A Talking Dictionary of Ainu: A New Version of Kanazawa's Ainu Conversational Dictionary. Available online: http://lah.soas.ac.uk/projects/ainu/ (accessed on 25 November 2015).
51. Jinbō, K.; Kanazawa, S. *Ainugo Kaiwa Jiten;* [Ainu conversational dictionary]; Kinkōdō Shoseki: Tokyo, Japan, 1898.
52. Nakagawa, H.; Bugaeva, A.; Kobayashi, M.; Kimura, K.; Akasegawa, S. Glossed Audio Corpus of Ainu Folklore. Available online: http://ainucorpus.ninjal.ac.jp (accessed on 10 December 2017).
53. Momouchi, Y.; Kobayashi, R. Dictionaries and Analysis Tools for the Componential Analysis of Ainu Place Name (in Japanese). *Eng. Res. Bull. Grad. Sch. Eng. Hokkai-Gakuen Univ.* **2010**, *10*, 39–49.
54. Chiba University Graduate School of Humanities and Social Sciences. Ainugo Mukawa Hōgen Nihongo—Ainugo Jiten [Japanese–Ainu dictionary for the Mukawa dialect of Ainu]. Available online: http://cas-chiba.net/Ainu-archives/index.html (accessed on 5 February 22017).
55. Nibutani Ainu Culture Museum. Ainu Language & Ainu Oral Literature. Available online: http://www.town.biratori.hokkaido.jp/biratori/nibutani/culture/language/ (accessed on 11 December 2017).
56. The Ainu Museum. Ainu-go Ākaibu [Ainu Language Archive]. Available online: http://ainugo.ainu-museum.or.jp/ (accessed on 15 August 2018).
57. Chiri, M. *Bunrui Ainu-go jiten. Dai-ikkan: shokubutsu-hen;* [dictionary of Ainu, vol. I: plants]; Nihon Jōmin Bunka Kenkyūsho: Tokyo, Japan, 1953.
58. Shigeru, K. *Kayano Shigeru no Ainugo Jiten;* [Shigeru Kayano's Ainu dictionary]; Sanseidō: Tokyo, Japan, 1996.
59. Bugaeva, A. Southern Hokkaido Ainu. In *The languages of Japan and Korea*; Tranter, N., Ed.; Routledge: London, UK, 2012; pp. 461–509.
60. Norvig, P. Natural language corpus data. In *Beautiful data*; Segaran, T.; Hammerbacher, J., Eds.; O'Reilly Media: Sebastopol, CA, USA, 2009; pp. 219–242.
61. Katayama, T. *Ainu shin-yōshū o yomitoku;* [studying the *Ainu shin-yōshū*]; Katayama Institute of Linguistic and Cultural Research: Tokyo, Japan, 2003.