Shrinking Japanese Morphological Analyzers With Neural Networks and Semi-supervised Learning

Arseny Tolmachev **Daisuke Kawahara** **Sadao Kurohashi**

Kyoto University, Graduate School of Informatics
Yoshida Honmachi, Sakyo-ku, Kyoto 606-8501, Japan
arseny@kotonoha.ws dk@i.kyoto-u.ac.jp kuro@i.kyoto-u.ac.jp

Abstract

For languages without natural word boundaries, like Japanese and Chinese, word segmentation is a prerequisite for downstream analysis. For Japanese, segmentation is often done jointly with part of speech tagging, and this process is usually referred to as morphological analysis. Morphological analyzers are trained on data hand-annotated with segmentation boundaries and part of speech tags. A segmentation dictionary or character n-gram information is also provided as additional inputs to the model. Incorporating this extra information makes models large. Modern neural morphological analyzers can consume gigabytes of memory. We propose a compact alternative to these cumbersome approaches which do not rely on any externally provided n-gram or word representations. The model uses only unigram character embeddings, encodes them using either stacked bi-LSTM or a self-attention network, and independently infers both segmentation and part of speech information. The model is trained in an end-to-end and semi-supervised fashion, on labels produced by a state-of-the-art analyzer. We demonstrate that the proposed technique rivals performance of a previous dictionary-based state-of-the-art approach and can even surpass it when training with the combination of human-annotated and automatically-annotated data. Our model itself is significantly smaller than the dictionary-based one: it uses less than 15 megabytes of space.

1 Introduction

Languages with a continuous script, like Japanese and Chinese, do not have natural word boundaries in most cases. Natural language processing for such languages requires to perform some variation of word segmentation.

Although some NLP applications, like neural machine translation, started to use unsupervised segmentation methods (Kudo and Richardson, 2018), resulting segmentation often has decisions which are not natural to humans. Supervised segmentation based on a human-defined standard is essential for applications which are designed for interaction on a word-level granularity, for example, full-text search. Segmentation is commonly done jointly with part of speech (**POS**) tagging and usually referred to as Morphological Analysis.

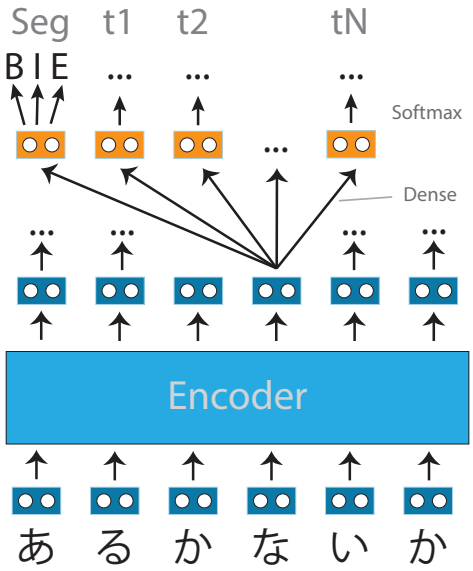

Figure 1: Proposed model. We encode character unigram embeddings into shared representations for each character. The shared representation is projected into a tag-specific representations from which we independently infer segmentation and per-character tags.

Modern Japanese Morphological Analyzers (**MA**) are very accurate, having a >99 segmentation tokenwise F1 score on news domain and a >98.5 F1 on web domain (Tolmachev et al., 2018). They often use segmentation dictionaries which define possible words. Also, their models are generally large and unwieldy, spanning hundreds of megabytes in case of traditional symbolic feature-based approaches. Neural models with word or n-

Proceedings of NAACL-HLT 2019, pages 2744–2755
Minneapolis, Minnesota, June 2 - June 7, 2019.

gram embeddings are even larger, easily reaching gigabytes. This makes it difficult to deploy MA in space-constrained environments such as mobile applications and browsers.

It has been shown that simple or straightforward models can match or outperform complex models when using a large number of training data. For example, a straightforward backoff technique rivals a complicated smoothing technique for language models (Brants et al., 2007). Pretraining a bidirectional language model on a large dataset helps to solve a variety of NLP tasks (Devlin et al., 2018). Our approach is inspired by this line of work.

Contributions We propose a very straightforward fully-neural morphological analyzer which uses *only character unigrams* as its input[1]. Such an analyzer, when trained only on human-annotated *gold* data has low accuracy. However, when trained on a large amount of automatically tagged *silver* data, the analyzer rivals and even outperforms, albeit slightly, the bootstrapping analyzer. We conclude that there is no need for rich input representation. Neural networks learn the information to combine characters into words by themselves when given enough data.

Ignoring explicit dictionary information and rich input representations makes it possible to make analyzers that are highly accurate and very compact at the same time. We also perform ablation experiments which show that the encoder component of such an analyzer is more important than character embeddings.

2 Morphological Analysis Overview

Segmentation is a cornerstone requirement for processing languages with a continuous script, and thus it has been studied for a long time. Most current approaches use either rich feature representation, e.g. character n-grams or their embeddings, or a segmentation dictionary. There exist two main lines of approaches: pointwise and search-based. Pointwise approaches make a segmentation decision for each character, usually based on the information from its surroundings. Search-based approaches look for a maximum scored interpretation in some structure over the input sentence.

Most Japanese analyzers use segmentation dictionaries which define corpus segmentation standards. They usually have rich POS information attached and are human-curated. One focus of segmentation dictionaries is to be consistent: it should be possible to segment a sentence using the dictionary entries only in a single correct way. Such dictionaries are often maintained together with annotated corpora. On the other hand, Chinese-focused systems do not put much focus on dictionaries. Still, almost all aproaches use rich feature templates or additional resources such as pretrained character n-gram or word embeddings, which increase the model size.

Pointwise approaches make a segmentation decision independently for each position. They can be seen as a sequence tagging task. Such approaches are more popular for Chinese.

KyTea (Neubig et al., 2011) is an example of this approach in Japanese. It makes a binary decision for each character: whether to insert a boundary before it or not. It can be seen as sequence tagging with {B, I} tagset. POS tagging is done after inferring segmentation. The decisions are made by feature-based approaches, using characters, character n-grams, character type information, and dictionary information as features. KyTea can use word features obtained from a dictionary. It checks whether the character sequence before and after the current character forms a word from the dictionary. It also checks whether the current word is inside a word.

Neural networks were shown to be useful for Japanese in this paradigm as well (Kitagawa and Komachi, 2017). They use character embeddings, character type embeddings, character n-gram embeddings, and tricks to incorporate dictionary information into the model.

Many studies on Chinese adopt the pointwise approach. Often, the segmentation task is reformulated as sequence tagging (Xue, 2003) with {B, I, E, S} tagset. Peng et al. (2004) showed that CRFs help further in this task. This tactic was followed by many subsequent feature-based approaches (Tseng et al., 2005; Zhao et al., 2006; Zhang et al., 2013), using character n-gram, character type and word features.

Neural networks were applied to this paradigm as well. Zheng et al. (2013) used a feed-forward network on character and categorical features that were shown to be useful for computing a segmentation score from a fixed window. Qi et al. (2014) used a similar architecture. They predicted not only segmentation but POS tags and performed

[1]The source code is avaliable at https://github.com/eiennohito/rakkyo

named entity recognition as well. The character representation was pretrained on a language modeling task. Shao et al. (2017) used a bidirectional recurrent network with GRU cells followed by a CRF layer for joint segmentation and POS tagging. They used pretrained character n-gram embeddings together with sub-character level information extracted by CNNs as features. Using a dictionary with NN is also popular (Zhang et al., 2018b; Liu et al., 2018).

Search-based approaches induce a structure over a sentence and perform a search over it. A most frequently used structure is a lattice which contains all possible segmentation tokens. The search then finds the highest scoring path through the lattice. Another branch of search-based approaches splits decisions into transitions (starting a new token and appending a character to the token) and searches for the highest scoring chain of transitions. This also can be seen as dynamically constructing a lattice while performing the search in it at the same time.

Lattice-based approaches are popular for the Japanese language. Most of the time, the lattice is based on words which are present in a segmentation dictionary and a rule-based component for handling out-of-dictionary words. Usually, there are no machine-learning components in lattice creation, but the scoring can be machine-learning based. We believe that the availability of high quality consistent morphological analysis dictionaries is the reason for that. Still, the work of Kaji and Kitsuregawa (2013) is a counterexample of a lattice-based approach for Japanese which uses a machine-learning component for creating the lattice.

Traditional lattice-based approaches for Japanese use mostly POS tags or other hidden information accessible from the dictionary to score paths through the lattice. JUMAN (Kurohashi, 1994) is one of the first analyzers, which uses a hidden Markov model with manually-tuned weights for scoring. Lattice path scores are computed using connection weights for each pair of part of speech tags.

Probably the most known and used morphological analyzer for Japanese is MeCab (Kudo et al., 2004), where CRFs were used for learning the scoring. MeCab is very fast: it can analyze almost 50k sentences per second. It also achieves acceptable accuracy, and so the tool is very popular. The speed is realized by precomputing feature weights, but it takes a lot of space when the total number of features gets large. For example, the UniDic model for modern Japanese v2.3.0[2] takes 5.5GB because it uses many feature templates.

There were studies which tried to integrate NN into lattice-based approaches as well. Juman++ (Morita et al., 2015) uses dictionary-based lattice construction with the combination of two models for path scoring: the feature-based linear model using soft-confidence weighted learning (Wang et al., 2016) and a recurrent neural network (Mikolov, 2012). It significantly reduced the number of both segmentation and POS tagging errors. However, it was very slow, being able to analyze only about 15 sentences per second, hence the original version was impractical. The following improvement (Tolmachev et al., 2018) greatly increased analysis speed by doing aggressive beam trimming and performing heavyweight NN evaluation only after lightweight scoring by the linear model.

Direct lattice-based approaches are not very popular for Chinese, but some are lattice-based in spirit. A line of work by Zhang and Clark (2008, 2010) builds the lattice dynamically from partial words, searching paths with a perceptron-based scorer and customized beam search. The dictionary is built dynamically from the training data as frequent word-tag pairs which help the system to prune unlikely POS tags for word candidates.

One more variation on lattice-based approaches for Chinese is the work by Cai and Zhao (2016). In this work, a segmentation dictionary is used to construct a subnetwork, which combines character representations into word representations used for computing sentence-wise segmentation scores. This can be seen as explicitly learning dictionary information by a model. Resulting segmentation is still created from the start to the end by growing words one by one while performing beam search. The follow up (Cai et al., 2017) simplifies that model and shows that greedy search can be enough for estimating segmentation when using neural networks. Still, this line of work does not consider POS tagging.

Transition-based approaches treat input data (most frequently – characters) as input queue and store a current, possibly incomplete, token in a buffer. Models usually infer whether they should create a new token from a character in the input

[2]https://unidic.ninjal.ac.jp/

queue or append an input character to the already existing token. Neural models are often used in this paradigm (Ma and Hinrichs, 2015; Zhang et al., 2016; Yang et al., 2017; Ma et al., 2018; Zhang et al., 2018a). Almost all of them use both word and charcter n-gram embeddings. This paradigm was extended to do parsing jointly with MA (Hatori et al., 2012; Kurita et al., 2017).

Semi-supervised approaches to segmentation and POS tagging fall into several categories. The first one uses raw or automatically-annotated data to precompute feature representations and then uses these feature representations for supervised learning. For example, Sun and Xu (2011) and Wang et al. (2011) use data from automatically segmented texts as features. They precomute the features beforehand and train an analyzer afterwards. In addition to that, Zhang et al. (2013) use a variation of smoothing for handling automatic annotation errors. A lot of neural-based methods pretrain word and character n-gram embeddings. Yang et al. (2017) pretrain a part of the model on different data sources, including automatically segmented text, but the model itself is trained only on the gold data.

Another approach is to use heterogeneous data (annotated in incompatible annotation standards). In addition to corpus statistics from a raw corpus, Zhao and Kit (2008) exploit heterogeneous annotations. Li et al. (2015) use corpora with different annotation standards. They combine tags into "bundles" (e.g. [NN, n]) and infer them at the same time while paying attention to ambiguity. Chen et al. (2016) train a classifier that can annotate several standards jointly.

Finally, it is possible to use raw or automatically-annotated data directly. A study (Suzuki and Isozaki, 2008) is an example of a feature-based algorithm which uses raw data. Tri-training (Zhou and Li, 2005) is a generic way to use raw data. They propose to train on automatically analyzed examples where two of three diverse analyzers agree. Søgaard (2010) show that tri-training helps English POS-tagging with SVM and MaxEnt-based approaches. Zhou et al. (2017) use self-training and tri-training for Chinese word segmentation. They, however, also pretrain other features like word-context character embeddings, chraracter unigrams and bigrams.

3 Proposed Approach

In order for MA to be practical, it should be not only accurate, but also fast and have relatively compact models. The speed of search-based approaches is dependent on how computationally heavy a weighting function is. Heavyweight models, like neural networks, require a large number of computations, and we think that it will be very difficult to create a practical search-based fully NN morphological analyzer with analysis speed comparable to traditional analyzers.

We do not want to use any explicit information about how to combine characters to form a word, like dictionaries, which takes space and is not trivial to incorporate into a character-based model. We also want our model to be fast, at least comparable with the speed of traditional analyzers. To this end, we follow a pointwise approach and force the neural network to learn the dictionary information from a corpus.

We use a straightforward architecture shown in Figure 1. We embed each character, and then apply an encoder, which produces an encoded representation for each character. Encoded character representations are independently transformed into tag representations. For each tag, the encoded representation is projected with a fully-connected layer with SeLU non-linearity (Klambauer et al., 2017). Finally, we multiply the tag representation by tag-specific embeddings and apply softmax non-linearity to get normalized tag probabilities.

Encoder Architectures We use two architectures for the encoder: a stacked bidirectional recurrent architecture with LSTM cells (Hochreiter and Schmidhuber (1997), **bi-LSTM**) and a Transofrmer-inspired mutihead self-attention network (Vaswani et al. (2017), **SAN**). We concatenate both directions of bi-LSTM outputs before passing them to the next layer without residual connections. We also apply layer normalization (Ba et al., 2016) to the concatenated outputs. We do not use dropout in encoders when using silver data for training.

Data Encoding Our model infers a tag for every input character. While this decision is natural for segmentation, POS tags are not usually tagged in this way.

For segmentation, we adopt {B, I, E} scheme. For POS tagging we broadcast tags to every character which is contained in a token. We use cor-

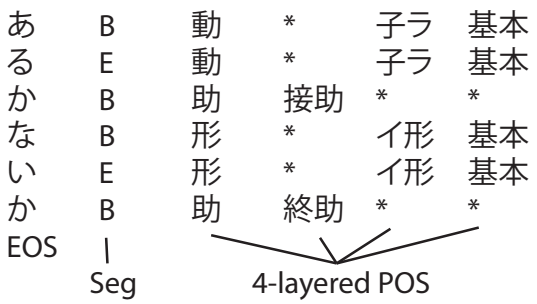

Figure 2: An example of full sentence annotation

あ	B	動	*	?	?
る	?	動	*	?	?
か	?	?	?	?	?
な	B	?	?	イ形	基本
い	E	?	?	イ形	基本
か	B	助	終助	*	*
EOS					

Figure 3: An example of partial sentence annotation

pora with the JUMAN-based segmentation standard (**Jumandic**), which has 4-layered POS tags: rough POS, fine POS, conjugation type and conjugation form. We treat each tag layer independently in our model, as shown in Figure 2.

We also consider a **partial annotation scheme**, where some tags are unknown. An example of partial sentence annotation is shown in Figure 3. Unknown tags are displayed by "?" symbols. We create partially annotated silver data by marking as unknown all tags which are ambiguous in a top-k analysis result. When computing the training loss, we treat unknown tags as padding: corresponding values are masked out of loss computation.

Loss Following Vaswani et al. (2017), we smooth softmax labels. They use the technique described by Szegedy et al. (2016), which uniformly distributes some small factor ϵ like 0.1 to incorrect labels. However, we do not induce a uniform smoothing. Instead, we want to prevent the model from being overconfident in its decisions without inducing uniformity. We slightly modify the cross-entropy loss as follows.

Remember that softmax probabilities are computed from unnormalized log-probabilities l_i as $q_i = e^{l_i}/Z$, where $Z = \sum_j e^{l_j}$. The cross-entropy loss will be $L = -\sum_i p_i \log q_i$, where p_i are gold probabilities. In our case the vector p is one-hot, meaning that $p_c = 1$ and other values are zero. This gives a sparse cross-entropy $L = -\log q_c = \log Z - l_c$, which is often implemented in deep learning frameworks. It has a minimum when $\log Z$ is equal to l_c, but it makes the model overconfident. Instead, we want to stop when $q_c = 1 - \epsilon$, or in other words $e^{l_c}/Z = 1 - \epsilon$. This gives us our modified loss:

$$L = \max(\log Z - l_c + \log(1 - \epsilon), 0).$$

It can be efficiently implemented using the sparse cross-entropy operation. In our experiments we use $\epsilon = 0.2$.

Our final loss is a weighted sum of individual tag softmax losses. We use a weight coefficient of 10 for segmentation and 2 for the first POS tag layer.

	Train		Test	
Corpus	Sents	Tokens	Sents	Tokens
KU	37k	930k	1783	46k
Leads	14k	217k	2195	36k

Table 1: Benchmark corpora sizes

4 Experiments

We conduct experiments on Japanese morphological analysis. For training we use two data sources. The first is usual human-annotated *gold* training data. The second is *silver* data from the results of automatic analysis. We use Juman++ V2 – the current state-of-the-art analyzer for the JUMAN segmentation standard as the bootstrap analyzer.

We use two gold corpora. The first is the Kyoto University Text Corpus (Kurohashi and Nagao (2003), referred to as **KU**), containing newspaper data. The second is the Kyoto University Web Document Leads Corpus (Hangyo et al. (2012), referred to as **Leads**) which consists of web documents. Corpus statistics are shown in Table 1. We denote models which use gold training data by **G**.

We take raw data to generate our silver annotated data from a crawled web corpus of 9.8B *unique* sentences. We sample 3B sentences randomly from it and analyze them using the Juman++ baseline model. From it we sample 500M sentences, which become our training silver data, prioritizing sentences which contain at least one not very frequent word. We prepare both top-scored (denoted as **T**) and non-ambigous in beam (denoted as **B**) variants of the silver data. Our silver data is in-domain for Leads and out-of-domain for KU.

Baselines We use four baselines: **JUMAN**, **MeCab**, **KyTea** and **Juman++** (V2). For MeCab,

Parameter	bi-LSTM	SAN
Char embedding size	128	128
Tag embedding size	32	32
# Layers	4	6
Hidden Size	128×2	32
# Heads	-	4
Projection Inner Dim	-	512
# Emedding Parameters	2.38M	2.38M
# Total Parameters	3.88M	3.59M

Table 2: Hyperparameters for neural models

KyTea and Juman++ we train a model using the same dictionary and merged training sections of KU and Leads, which is evaluated on each corpus independently.

Neural Models The hyper-parameters of the bi-LSTM-based model are displayed in Table 2. We use all unique characters present in our huge web corpus (18,581) as input. We select sizes of both neural models restricting the total number of parameters to be less than 4M. For optimization we use the Adam optimizer (Kingma and Ba, 2016) with hyperparameters and learning rate scheduling described by Vaswani et al. (2017). We train all models on Nvidia GPUs. On a single GeForce 1080Ti the bi-LSTM model can consume about 4,500 sentences per second and the SAN-based model about 6,500 sentences per second for training. We denote bi-LSTM-based models by **L** and SAN-based models by **S** in experimental results.

Treatment of Gold Data Existing methods are already highly accurate on this task, and it is difficult to perform hyperparameter and architecture selection reliably with a small development set. Because of that, we split our data in an unusual way. Generally, we use the silver data (B or T) as a train set, the human-annotated original training data (G) as a dev set and the original test set as a test set. Our hyperparameter selection decisions were based entirely on this setting. We do not perform additional hyperparameter search for a combination of silver and gold data for training.

The exception is cases when we use *only gold data* for training. For that, we cheat and optimize our hyperparameters, including dropout, which we use only for this setting, on test scores. Nonetheless, the best scores on this setting are significantly lower than the worst baseline.

	KU, News			Leads, Web		
Analyzer	Seg	+P1	+P2	Seg	+P1	+P2
			Baselines			
JUMAN	98.41	97.18	95.45	98.09	96.96	95.71
MeCab	99.10	98.56	97.59	98.25	97.60	96.22
KyTea	99.13	98.25	97.01	97.98	96.85	95.11
Juman++	**99.52**	**99.10**	97.86	98.61	98.07	96.70
			bi-LSTM			
L:G	97.46	96.56	94.78	96.33	95.43	93.46
L:B	99.22	98.82	97.50	98.57	98.01	96.61
L:T	99.33	98.90	97.59	98.68	98.16	96.71
L:BG	99.43	99.05	**98.06**	98.59	98.04	96.76
L:TG	99.43	99.05	98.01	**98.71**	**98.19**	96.80
			Self-attention			
S:G	98.28	97.67	95.66	97.23	96.36	93.91
S:B	99.19	98.75	97.34	98.56	97.99	96.59
S:T	99.23	98.78	97.36	98.66	98.15	96.75
S:BG	99.30	98.90	97.83	98.60	98.03	96.70
S:TG	99.37	98.97	97.93	**98.70**	98.15	**96.83**
			Pre-training scenario			
S:B→G(a)	99.24	98.85	97.75	98.58	98.03	96.64
S:B→G(b)	99.15	98.65	97.55	98.39	97.78	96.36
S:B→G(c)	99.27	98.82	97.75	98.50	97.91	96.43
S:B→G(d)	99.26	98.82	97.76	98.52	97.94	96.48

Table 3: Test F1 score comparison on benchmark corpora. Legend: bi-[L]STM, [S]AN, [G]old data, [T]op-only and [B]eam-non-ambigous silver data.

Experimental Results Results of our experiments are shown in Table 3. For each analyzer, we show six values. **Seg** is a tokenwise F1 measure on segmentation. **+P1** requires the 1st layer of POS tags (coarse-grained POS tags) also to match gold data. For the sake of simplicity, we use only POS tags co-located with "B" Seg tags for the evaluation. **+P2** is analogous for the 2nd layer of POS tags. For all results in this table, we train NN-based models for a single epoch, which means *the training procedure sees each silver sentence only once*. We use one gold example for ten silver examples for mixed-data settings, looping over the gold data until the silver data is extinguished.

Training neural models only on gold data quickly results in overfitting which can be seen in L:G and S:G results. These scores are significantly lower than that of our worst baseline: JUMAN.

Models trained on only non-ambiguous silver data (*:B) are comparable to the best baseline on Leads (in-domain), although they cannot reach the accuracy of Juman++ on KU. Using top-only silver data (*:T) further improves accuracy. Both of our models in this setting slightly outperform previous Leads SOTA and have more or less the same accuracy. On KU, the LSTM-based model seems to be slightly better than the SAN-based one. In the context of semi-supervised learning, tri-training emphasizes using data when there exists a disagreement between the analyzers. Instead, we throw

	Size, MB		
Analyzer	Dictionary	Model	Total
JUMAN	288	1	289
MeCab	312	8	320
KyTea:G	-	569	569
KyTea:TG	-	3218	3218
Juman++	157	288	434
bi-LSTM	1	14	15
SAN	1	13	14

Table 4: MA model sizes for Jumandic

away difficult cases for beam-based data, denoising it in a sense, but NN seem to handle that kind of noise relatively well.

Adding the gold data to the silver data (*:BG, *:TG) allows both models to improve their accuracy further. Results on Leads are comparable for both L:TG and S:TG and higher than the previous SOTA, giving segmentation error reduction of 8% in comparison to Juman++. On KU, the LSTM-based models seem to perform better without a significant difference on the TG and BG settings, while still underperforming the Juman++ baseline except +P2 case, where both models are stronger than Juman++.

Pre-training Scenario We also check the fine-tuning approach when we first learn the representations on a large corpus and then refine the model on a gold corpus. S:B→G(a-d) are four such runs of a SAN-based model with different hyperparameters. All four runs are initialized with the same S:B model and trained on the gold data only. We found it difficult to find good hyperparameters for fine-tuning. The models were prone to overfit very fast. Mixing gold and silver data resulted in stable training without hyperparameter search.

Model Sizes We compare the model sizes of analyzers in Table 4. In case of dictionary-based analyzers the dictionary takes most of the space. We count sizes of compiled models for all analyzers. KyTea, as another example of pointwise MA, uses string-based features and treats its features uniformly, hence dictionary size is not applicable to it. A **KyTea:TG** variant that uses additional 2M silver sentences takes almost 6x the space of the original model, reaching 3GB. When using neural networks, on the other hand, it is possible to control model sizes more easily. Moreover, our proposed models take significantly less space while having comparable accuracy.

Analyzer	KU	Leads
KyTea-D:G	98.45	97.04
KyTea-D:T	98.51	98.10
KyTea-D:TG	99.18	98.31
KyTea:G	99.13	97.98
KyTea:TG	99.33	98.42

Table 5: KyTea test Seg F1 comparison. -D models do not use the dictionary. T models use silver data (2M sentences, created like in the main experiment)

Dictionary-based analyzers store other information, like readings and lemma forms, in addition to token surface forms and POS, but removing that information would not make model sizes comparable with NN-based ones. For NN-based analyzers, we count a dictionary as 1 MB because they need a character-to-id mapping to work. However, the list of characters contains non-frequently used characters, some of which could be treated as UNKs without any accuracy loss. We also treat weights as 4-byte floating points, and so it would be possible to further decrease the NN model size, for example by using less precise storage formats.

5 Discussion

Dictionaries Dictionary information is usually added to character-based models either using a binary feature vector (e.g. a dictionary contains a trigram to the left of the decision point) or word embeddings. We believe that a dictionary can be replaced with a large training corpus which includes most of the entries from that dictionary. A neural model with only the unigram character input can solve word segmentation and POS tagging only if it builds some knowledge about the dictionary internally. Our main experimental results (Table 4) show that it seems to be the case and there is no need to model the dictionary explicitly.

Table 5 shows an effect of using dictionaries and silver data on KyTea, an instance of symbolic feature-based analyzer. Models tagged with **T** use additional 2M silver training data analyzed by Juman++. KyTea has better accuracy in settings when it uses the dictionary. The dictionary even helps in the setting with additional silver data. Unfortunately, the model size increases as well, limiting the amount of silver data we can use, and the accuracy cannot rival neural approaches.

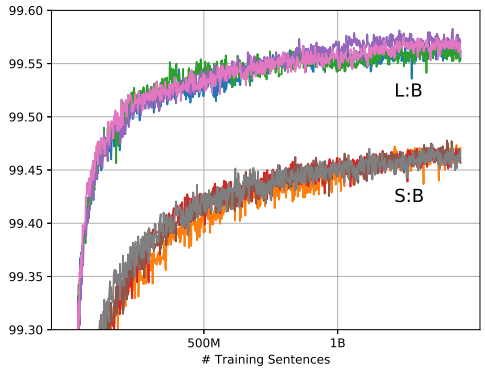

Figure 4: Dev Seg F1 curves for L:B and S:B

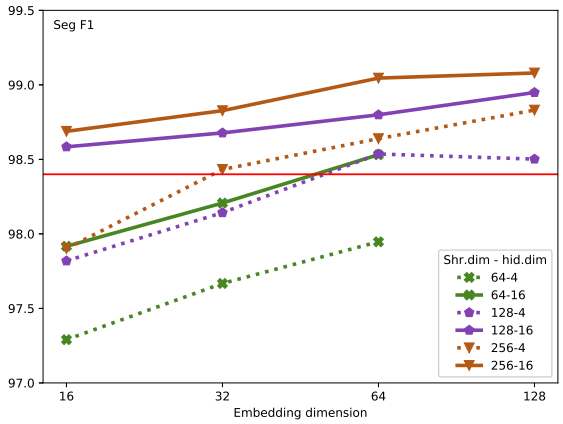

Figure 5: Effect of embedding size on Seg F1

How much data do we need? For our main experiments, we train all models for a single epoch on our silver dataset. Figure 4 shows KU train (our dev set) Seg F1 curves for L:B and S:B for three epochs. We ran each experiment four times with different random seeds. The learning curves become less sloppy when reaching 500M sentences but do not become flat there. The training does not seem to completely converge even after 3 epochs. We still use one full epoch (500M) for our main experiments. The curves are pretty noisy, but it seems that the model is robust with respect to initialization.

SAN Ablation Experiments The proposed MA achieves high accuracy while having very compact models. The inputs do not contain any information on how to combine characters into words and we assume that the model learns it from the data. To get the model size even smaller, we check which model parts contribute more to the resulting analysis accuracy, meaning that they contain the dictionary knowledge.

We perform ablation experiments on the SAN model by varying its hyperparameters and checking how it affects the accuracy of the resulting analyzer. The LSTM model could not converge in this setting. We used 2.5M of silver training data for these experiments.

Figure 5 shows the segmentation F1 score when varying input embedding, shared representation and SAN hidden dimension sizes. JUMAN score, as a lowest acceptable baseline, is shown in red. The embedding size seems to have a lower impact on accuracy than the shared representation and the SAN hidden dimension size. Namely, the (128-16) model with the embedding size of 16 has higher accuracy than the (128-4) model with the embedding size of 128. Accordingly, we believe that the encoder contributes much stronger to learning the dictionary than character embeddings.

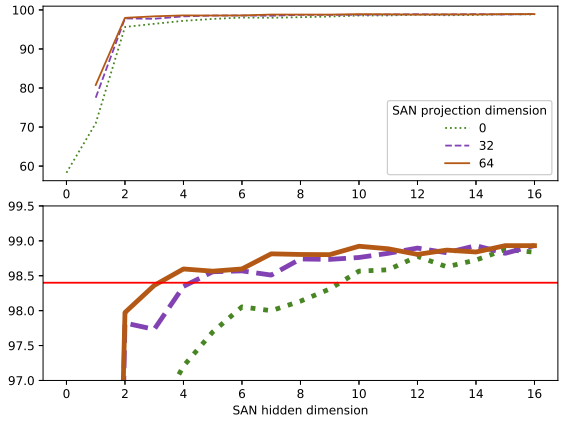

Figure 6: Effect of SAN hidden dimension on Seg F1

One more interesting observation is that the models are still better than JUMAN, while having much less parameters than our base model. We explore more extreme settings of the SAN hidden state, shown in Figure 6. We fix embedding and shared representation dimensions to 128 and vary the SAN hidden and projection dimensions. The lower subgraph is a scale-up version of top graph. The point at SAN hidden size equal to 0 means that we directly use unigram embeddings to predict segmentation without any encoder.

The SAN projection size is consistent with accuracy, especially on smaller SAN hidden sizes. An interesting observation here is that the SAN model seems to work even with hidden dimension of 2. When the hidden dimension size reaches 4, the extremely small model accuracy is higher than the JUMAN baseline. This shows that it is possible to create an extremely small MA with acceptable accuracy.

System Segmentation	Correct Segmentation	Transliteration	Meaning
こう|ゆう|曲って	こうゆう|曲|って	ko:yu: kyoku tte	this song is
なんて|すん|ごい	なんて|すんごい	nante sungoi	how awesome
んな|わけな|いって	んな|わけ|ない|って	nna wake nai tte	no way!
あっ|ちゃんと|遊び|たい	あっ|ちゃん|と|遊び|たい	*see main text*	

Table 6: n-grams with inconsistent POS tags which are also Juman++ errors

Label Uncertainty and Error Analysis Because our neural models infer all tags independently, they can be inconsistent, for example, a word can have different POS tags on different characters. We looked into frequent 3-grams where the central word has inconsistent tags (POS tags are not the same for all characters, or they do not form a correct 4-layered tag). Most of these trigrams occur in ambiguous situations.

We have picked several examples which are actually errors in Juman++ segmentation as well. They are shown in Table 6. In Japanese, words often have several orthographic forms. The most common variant is usage of hiragana (phonetic script) instead of kanji (ideographic characters). Verbs can have different possible endings, e.g. 曲がる and 曲る (magaru – to turn or bend) are two orthographic variants of a single verb. There are also colloquial variants; namely the verb 言う is usually read as いう (iu – to say), but can also be written as ゆう because the pronunciation is close. These phenomena are relatively common in web and user-generated texts, but corpus and segmentation dictionary coverage of them is not very good.

The first two examples contain alternative colloquial spellings of words こういう (ko:iu – such) and すごい (sugoi – awesome). In the first example the system incorrectly recognizes 曲\|って (kyoku tte) as 曲って (magatte) – a conjugation of 曲る. The fourth example (a chanto asobitai/ac-chan to asobitai - ah! [I] want to play properly/[I] want to play with ac-chan <person name>) is actually ambiguous and can have two meanings. The second one is more probable though. The fact that frequent words with uncertain POS tags are Juman++ errors as well implies that insufficient gold data causes the uncertainty.

We also compare differences between Juman++ and our models to get an insight on general problems with proposed methods. Neural models make many errors in hiragana words. For example, both neural models make errors in the sentence 弱者\|が\|とうた\|さ\|れて (jyakusya ga to:ta sarete - weaklings lose to natural selection). LSTM makes a segmentation mistake (と\|うたさ) and SAN does a POS tagging mistake, while Juman++ produces the correct answer. It knows that とうた is a special type of noun that is often followed by されて from POS tags. Hiragana-based spellings of most content words are somewhat rare in Japanese, and NN models do not have enough training data for these spellings. It could be possible to improve the situation by using data augmentation techniques. Another frequent problem is segmentation and tagging of proper nouns. We believe that this problem could be solved by data augmentation, but we leave this as future work.

6 Conclusion and Future Work

We presented a novel way to train small neural models for Japanese Morphological analysis by directly feeding the network a large number of silver training data. Our method achieves new SOTA on web domain when combining the silver data with gold one. This is an empirical evidence that there is no need for feature engineering for neural morphological analysis at all. A neural network can learn implicit dictionary information itself and it does not need to be large. We also show that training by mixing the data together works better than fine-tuning and is more stable.

Our work can be extended in the future in different ways. We will consider how to make the model to recognize new words, which is an important feature for a practical analyzer. Using tri-training also seems to be a natural extension for this work. It is easy to provide diverse models, required for tri-training, by using different types of encoder and varying network parameters. Furthermore, our tagging approach should be universal and work with other tasks like named entity recognition. A method to incorporate tags with a large number of possible values (like readings and lemmas) without introducing embeddings for them, hence keeping the models small, could also be a useful extension.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization. *arXiv:1607.06450 [cs]*.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. 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)*, pages 858–867, Prague, Czech Republic. Association for Computational Linguistics.

Deng Cai and Hai Zhao. 2016. Neural Word Segmentation Learning for Chinese. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 409–420, Berlin, Germany. Association for Computational Linguistics.

Deng Cai, Hai Zhao, Zhisong Zhang, Yuan Xin, Yongjian Wu, and Feiyue Huang. 2017. Fast and Accurate Neural Word Segmentation for Chinese. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 608–615, Vancouver, Canada. Association for Computational Linguistics.

Hongshen Chen, Yue Zhang, and Qun Liu. 2016. Neural Network for Heterogeneous Annotations. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 731–741, Austin, Texas. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*. ArXiv: 1810.04805.

Masatsugu Hangyo, Daisuke Kawahara, and Sadao Kurohashi. 2012. Building a Diverse Document Leads Corpus Annotated with Semantic Relations. In *Proceedings of the 26th Pacific Asia Conference on Language, Information, and Computation*, pages 535–544, Bali, Indonesia. Faculty of Computer Science, Universitas Indonesia.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2012. Incremental Joint Approach to Word Segmentation, POS Tagging, and Dependency Parsing in Chinese. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1045–1053, Jeju Island, Korea. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780.

Nobuhiro Kaji and Masaru Kitsuregawa. 2013. Efficient Word Lattice Generation for Joint Word Segmentation and POS Tagging in Japanese. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 153–161, Nagoya, Japan. Asian Federation of Natural Language Processing.

Diederik P. Kingma and Jimmy Ba. 2016. Adam: A method for stochastic optimization. *arXiv:1412.6980 [cs.LG]*.

Yoshiaki Kitagawa and Mamoru Komachi. 2017. Long Short-Term Memory for Japanese Word Segmentation. *arXiv:1709.08011 [cs]*. ArXiv: 1709.08011.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-Normalizing Neural Networks. In *Advances in Neural Information Processing Systems 30*, pages 971–980. Curran Associates, Inc.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying Conditional Random Fields to Japanese Morphological Analysis. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

Shuhei Kurita, Daisuke Kawahara, and Sadao Kurohashi. 2017. Neural Joint Model for Transition-based Chinese Syntactic Analysis. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1204–1214, Vancouver, Canada. Association for Computational Linguistics.

Sadao Kurohashi. 1994. Improvements of Japanese morphological analyzer JUMAN. In *Proceedings of The International Workshop on Sharable Natural Language, 1994*.

Sadao Kurohashi and Makoto Nagao. 2003. Building A Japanese Parsed Corpus. In *Treebanks: Building and Using Parsed Corpora*, Text, Speech and Language Technology, pages 249–260. Springer Netherlands, Dordrecht.

Zhenghua Li, Jiayuan Chao, Min Zhang, and Wenliang Chen. 2015. Coupled Sequence Labeling on Heterogeneous Annotations: POS Tagging as a Case Study. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1783–1792, Beijing, China. Association for Computational Linguistics.

Junxin Liu, Fangzhao Wu, Chuhan Wu, Yongfeng Huang, and Xing Xie. 2018. Neural Chinese Word Segmentation with Dictionary Knowledge. In *Natural Language Processing and Chinese Computing*, Lecture Notes in Computer Science, pages 80–91. Springer International Publishing.

Ji Ma, Kuzman Ganchev, and David Weiss. 2018. State-of-the-art Chinese Word Segmentation with Bi-LSTMs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4902–4908, Brussels, Belgium. Association for Computational Linguistics.

Jianqiang Ma and Erhard Hinrichs. 2015. Accurate Linear-Time Chinese Word Segmentation via Embedding Matching. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1733–1743, Beijing, China. Association for Computational Linguistics.

Tomas Mikolov. 2012. *Statistical Language Models Based on Neural Networks*. Ph. D. Thesis, Brno University of Technology, Brno.

Hajime Morita, Daisuke Kawahara, and Sadao Kurohashi. 2015. Morphological Analysis for Unsegmented Languages using Recurrent Neural Network Language Model. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2292–2297, Lisbon, Portugal. Association for Computational Linguistics.

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise Prediction for Robust, Adaptable Japanese Morphological Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 529–533, Portland, Oregon, USA. Association for Computational Linguistics.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese Segmentation and New Word Detection Using Conditional Random Fields. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yanjun Qi, Sujatha G. Das, Ronan Collobert, and Jason Weston. 2014. Deep Learning for Character-Based Information Extraction. In *Advances in Information Retrieval*, Lecture Notes in Computer Science, pages 668–674. Springer International Publishing.

Yan Shao, Christian Hardmeier, Jörg Tiedemann, and Joakim Nivre. 2017. 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 (Volume 1: Long Papers)*, pages 173–183, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Weiwei Sun and Jia Xu. 2011. Enhancing Chinese Word Segmentation Using Unlabeled Data. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 970–979, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Jun Suzuki and Hideki Isozaki. 2008. Semi-Supervised Sequential Labeling and Segmentation Using Giga-Word Scale Unlabeled Data. In *Proceedings of ACL-08: HLT*, pages 665–673, Columbus, Ohio. Association for Computational Linguistics.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Anders Søgaard. 2010. Simple Semi-Supervised Training of Part-Of-Speech Taggers. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 205–208, Uppsala, Sweden. Association for Computational Linguistics.

Arseny Tolmachev, Daisuke Kawahara, and Sadao Kurohashi. 2018. Juman++: A Morphological Analysis Toolkit for Scriptio Continua. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 54–59. Association for Computational Linguistics.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A Conditional Random Field Word Segmenter for Sighan Bakeoff 2005. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Jialei Wang, Peilin Zhao, and Steven C. H. Hoi. 2016. Soft Confidence-Weighted Learning. *ACM Trans. Intell. Syst. Technol.*, 8(1):15:1–15:32.

Yiou Wang, Jun'ichi Kazama, Yoshimasa Tsuruoka, Wenliang Chen, Yujie Zhang, and Kentaro Torisawa. 2011. Improving Chinese Word Segmentation and POS Tagging with Semi-supervised Methods Using Large Auto-Analyzed Data. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 309–317, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Nianwen Xue. 2003. Chinese Word Segmentation as Character Tagging. In *International Journal of Computational Linguistics & Chinese Language Process-*

ing, Volume 8, Number 1, February 2003: Special Issue on Word Formation and Chinese Language Processing, pages 29–48.

Jie Yang, Yue Zhang, and Fei Dong. 2017. Neural Word Segmentation with Rich Pretraining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 839–849, Vancouver, Canada. Association for Computational Linguistics.

Longkai Zhang, Houfeng Wang, Xu Sun, and Mairgup Mansur. 2013. Exploring Representations from Unlabeled Data with Co-training for Chinese Word Segmentation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 311–321, Seattle, Washington, USA. Association for Computational Linguistics.

Meishan Zhang, Nan Yu, and Guohong Fu. 2018a. A Simple and Effective Neural Model for Joint Word Segmentation and POS Tagging. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(9):1528–1538.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2016. Transition-Based Neural Word Segmentation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 421–431, Berlin, Germany. Association for Computational Linguistics.

Qi Zhang, Xiaoyu Liu, and Jinlan Fu. 2018b. Neural Networks Incorporating Dictionaries for Chinese Word Segmentation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5682–5689.

Yue Zhang and Stephen Clark. 2008. Joint Word Segmentation and POS Tagging Using a Single Perceptron. In *Proceedings of ACL-08: HLT*, pages 888–896, Columbus, Ohio. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2010. A Fast Decoder for Joint Word Segmentation and POS-Tagging Using a Single Discriminative Model. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 843–852, Cambridge, MA. Association for Computational Linguistics.

Hai Zhao, Chang-Ning Huang, Mu Li, and Bao-Liang Lu. 2006. 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*. Association for Computational Linguistics.

Hai Zhao and Chunyu Kit. 2008. Exploiting unlabeled text with different unsupervised segmentation criteria for chinese word segmentation. *Research in Computing Science*, 33:93–104.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep Learning for Chinese Word Segmentation and POS Tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 647–657, Seattle, Washington, USA. Association for Computational Linguistics.

Hao Zhou, Zhenting Yu, Yue Zhang, Shujian Huang, XIN-YU DAI, and Jiajun Chen. 2017. Word-Context Character Embeddings for Chinese Word Segmentation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 760–766, Copenhagen, Denmark. Association for Computational Linguistics.

Zhi-Hua Zhou and Ming Li. 2005. Tri-training: exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1529–1541.