# Japanese Sentence Compression with a Large Training Dataset

**Shun Hasegawa**
Tokyo Institute of Technology, Japan
hasegawa.s@lr.pi.titech.ac.jp

**Yuta Kikuchi**
Preferred Networks, Inc., Japan
kikuchi@preferred.jp

**Hiroya Takamura**
Tokyo Institute of Technology, Japan
takamura@pi.titech.ac.jp

**Manabu Okumura**
Tokyo Institute of Technology, Japan
oku@pi.titech.ac.jp

## Abstract

In English, high-quality sentence compression models by deleting words have been trained on automatically created large training datasets. We work on Japanese sentence compression by a similar approach. To create a large Japanese training dataset, a method of creating English training dataset is modified based on the characteristics of the Japanese language. The created dataset is used to train Japanese sentence compression models based on the recurrent neural network.

## 1 Introduction

Sentence compression is the task of shortening a sentence while preserving its important information and grammaticality. Robust sentence compression systems are useful by themselves and also as a module in an extractive summarization system (Berg-Kirkpatrick et al., 2011; Thadani and McKeown, 2013). In this paper, we work on Japanese sentence compression by deleting words. One advantage of compression by deleting words as opposed to abstractive compression lies in the small search space. Another one is that the compressed sentence is more likely to be free from incorrect information not mentioned in the source sentence.

There are many sentence compression models for Japanese (Harashima and Kurohashi, 2012; Hirao et al., 2009; Hori and Furui, 2004) and for English (Knight and Marcu, 2000; Turner and Charniak, 2005; Clarke and Lapata, 2006). In recent years, a high-quality English sentence compression model by deleting words was trained on a large training dataset (Filippova and Altun, 2013; Filippova et al., 2015). While it is impractical to create a large training dataset by hand, one can be created automatically from news articles (Filippova and Altun, 2013). The procedure is as follows (where $S$, $H$, and $C$ respectively denote the first sentence of an article, the headline, and the created compressed sentence of $S$). Firstly, to restrict the training data to grammatical and informative sentences, only news articles satisfying certain conditions are used. Then, nouns, verbs, adjectives, and adverbs (i.e., content words) shared by $S$ and $H$ are identified by matching word lemmas, and a rooted dependency subtree that contains all the shared content words is regarded as $C$.

However, their method is designed for English, and cannot be applied to Japanese as it is. Thus, in this study, their method is modified based on the following three characteristics of the Japanese language: (a) Abbreviation of nouns and nominalization of verbs frequently occur in Japanese. (b) Words that are not verbs can also be the root node especially in headlines. (c) Subjects and objects that can be easily estimated from the context are often omitted.

The created training dataset is used to train three models. The first model is the original Filippova et al.'s model, an encoder-decoder model with a long short-term memory (LSTM), which we extend in this paper to make the other two models that can control the output length (Kikuchi et al., 2016), because controlling the output length makes a compressed sentence more informative under the desired length.

## 2 Creating training dataset for Japanese

Filippova et al.'s method of creating training data consists of the conditions imposed on news articles and the following three steps: (1) identification of shared content words, (2) transformation of a dependency tree, and (3) extraction of the min-



*Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Short Papers)*, pages 281–286
Vancouver, Canada, July 30 - August 4, 2017. 
https://doi.org/10.18653/v1/P17-2044


Figure 1: Dependency tree of $S$[1]. Squares and arrows respectively indicate chunks and dependency relations. Bold arrows mean that chunks are merged into a single node. Bold squares are extracted as $C$.

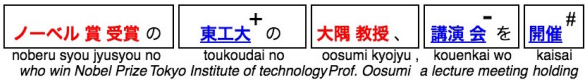


Figure 2: A sequence of chunks of $H$[1]

imum rooted subtree. We modified their method based on the characteristics of the Japanese language as follows. To explain our method, a dependency tree of $S$ and a sequence of *bunsetsu* chunks of $H$ in Japanese are shown in Figures 1 and 2. Note that nodes of dependency trees in Japanese are *bunsetsu* chunks each consisting of content words followed by function words.

### 2.1 Identification of shared content words

Content words shared by $S$ and $H$ are identified by matching lemmas and pronominal anaphora resolution in Filippova et al.'s method. Abbreviation of nouns and nominalization of verbs frequently occur in Japanese (characteristic (a) in Introduction), and it is difficult to identify these transformations simply by matching lemmas. Thus, after the identification by matching lemmas, two identification methods (described below) using character-level information are applied. Note that pronominal anaphora resolution is not used, because pronouns are often omitted in Japanese.

**Abbreviation of nouns:** There are two types of abbreviations of nouns in Japanese. One is the abbreviation of a proper noun, which shortens the form by deleting characters (e.g., the pair with "+" in Figures 1 and 2). The other is the abbreviation of consecutive nouns, which deletes nouns that behave as adjectives (e.g., the pair with "-"). To deal with such cases, if the character sequence of the noun sequence in a chunk in $H$ is identical to a subsequence of characters composing the noun sequence in a chunk in $S$, the noun sequences in $H$ and $S$ are regarded as shared.

**Nominalization of verbs:** Many verbs in Japanese have corresponding nouns with similar meanings (e.g., the pair with # in Figures 1 and 2). Such pairs often share the same Chinese character, *kanji*. Kanji is an ideogram and is more informative than the other types of Japanese letters. Thus, if a noun in $H$ and a verb in $S$[2] share one kanji, the noun and the verb are regarded as shared.

### 2.2 Transformation of a dependency tree

Some edges in dependency trees cannot be cut without changing the meaning or losing the grammaticality of the sentence. In Filippova et al.'s method, the nodes linked by such an edge are merged into a single node before extraction of the subtree. The method is adapted to Japanese as follows. If the function word of a chunk is one of the specific particles[3], which often make obligatory cases, the chunk and its modifiee are merged into a single node. In Figure 1, the chunks at the start and end of a bold arrow are merged into a single node.

### 2.3 Extraction of the minimum rooted subtree

In Filippova et al.'s method, the minimum rooted subtree that contains all the shared content words is extracted from the transformed tree. We modify their method to take into account the characteristics (a) and (b).

**Deleting the global root:** In English, only verbs can be the root node of a subtree. However, in Japanese, words with other parts-of-speech can also be the root node in headlines (characteristics (b)). Therefore, the global root, which is the root node of $S$, and the chunks including a word that can be located at the end of a sentence[4], are the candidates for the root node of a subtree. Then, if the root node is not the global root, words succeeding the word that can be located at the end are removed from the root node. In Figure 1, among the two words with "*" that can be located at the

[1] Words in red are regarded as shared through lemma matching, while words in blue (also underlined) are through other ways.

[2] Candidates are restricted to verbs consisting of one kanji and some following hiragana.

[3] "を (wo)", "に (ni)", "が (ga)" and "は (ha)"

[4] An auxiliary verb, a noun, or a verb of which next word is not a noun, an auxiliary verb, or "の (no)"

end, the latter is extracted as the root, and the succeeding word is removed from the chunk.

**Reflecting abbreviated forms:** Abbreviation of nouns frequently occurs in Japanese (characteristic (a)). Thus, in *C*, original forms are replaced by their abbreviated forms obtained as explained in Section 2.1 (e.g., the pair with "-" in Figures 1 and 2). However, we do not allow the head of a chunk to be deleted to keep the grammaticality. We also restrict here ourselves to word-level deletion and do not allow character-level deletion, because our purpose is to construct a training dataset for compression by word deletion. In the example of Figure 1, chunks in bold squares are extracted as *C*.

### 2.4 Conditions imposed on news articles

Filippova et al.'s method imposed eight conditions on news articles to restrict the training data to grammatical and informative sentences. In our method, these conditions are modified to adapt to Japanese. Firstly, the condition "*S* should include content words of *H* in the same order" is removed, because word order in Japanese is relatively free. Secondly, the condition "*S* should include all content words of *H*" is relaxed to "the ratio of shared content words to content words in *H* is larger than threshold $\theta$" because in Japanese, subjects and objects that can be easily estimated from the context are often omitted (characteristics (c)). In addition, two conditions "*H* should have a verb" and "*H* must not begin with a verb" are removed, leaving four conditions[5].

## 3 Sentence compression with LSTM

Three models are used for sentence compression. Each model predicts a label sequence, where the label for each word is either "retain" or "delete" (Filippova et al., 2015). The first is an encoder-decoder model with LSTM (*lstm*) (Sutskever et al., 2014). Given an input sequence $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, $y_t$ in label sequence $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ is computed as follows:

$$s_0 = encoder(\mathbf{x})$$
$$(s_t, m_t) = decoder(s_{t-1}, m_{t-1}, e_1(x_t) \oplus e_l(y_{t-1}))$$
$$y_t = softmax(W * s_t + b),$$

[5]"*H* is not a question", "both *H* and *S* are not less than four words.", "*S* is more than 1.5 times longer than *H*", and "*S* is more than 1.5 times longer than *C*".

| threshold $\theta$ | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1.0 |
|---|---|---|---|---|---|---|---|---|---|---|
| size(10k) | 202 | 189 | 172 | 154 | 134 | 106 | 81 | 59 | 35 | 27 |
| vocab(k) | 105 | 102 | 98 | 94 | 90 | 81 | 72 | 62 | 48 | 42 |
| ROUGE-2 | 54.6 | 54.4 | 54.0 | 54.3 | 55.3 | 54.0 | 55.1 | 54.4 | 53.0 | 53.3 |

Table 1: Created training data with each $\theta$. ROUGE-2 is the score of compressed sentence generated by the model trained on each training data to target.

where $\oplus$ indicates concatenation, $s_t$ and $m_t$ are respectively a hidden state and a memory cell at time $t$, and $e_1(word)$ is embedding of $word$. If $label$ is "retain", $e_l(label)$ is $(1; 0)$; otherwise, $(0; 1)$. $m_0$ is a zero vector.

As the second and the third models, we extend the first model to control the output length (Kikuchi et al., 2016). The second model, *lstm+leninit*, initializes the memory cell of the decoder as follows: $m_0 = tarlen * b_{len}$ where $tarlen$ is the desired output length, and $b_{len}$ is a trainable parameter. The third model, *lstm+lenemb*, uses the embedding of the potential desired length $e_2(length)$ as an additional input. In this case, $e_1(x_t) \oplus e_l(y_{t-1}) \oplus e_2(l_t)$ is used as the input of the decoder where $l_t$ is the potential desired length at time $t$.

## 4 Experiments

The created training datasets were used to train three models for sentence compression. To see the effect of the modified subtree extraction method (Section 2.3), two training datasets were tested: *rooted* and *multi-root+*. *rooted* includes only deletion of the leafs in a dependency tree. In contrast, *multi-root+* includes deleting the global root and reflecting abbreviated forms besides it.

**Setting:** Training datasets were created from seven million, 35-years' worth of news articles from the Mainichi, Nikkei, and Yomiuri newspapers, from which duplicate sentences and sentences in test data were filtered out. Gold-standard data were composed of the first sentences of the 1,000 news articles from Mainichi, each of which has 5 compressed sentences separately created by five human annotators. 100 sentences of the gold-standard data were used as development data, while the other sentences were used as test data. The three models were trained on datasets created with each value of threshold $\theta$[6], which is the parameter used in the condition (introduced in Sec-

[6]In Table 2, the thresholds for our models are 0.7, 0.5, 0.3, 0.6, 0.4, and 0.2, respectively, from the top.

| dataset | model | R-1 | ROUGE-2 | | | R-L | CR |
|---|---|---|---|---|---|---|---|
| | | | R(R-2) | P | F | | |
| | *prop-w-dpnd* | 73.8 | 56.5 | 51.5 | 53.7 | 32.7 | 60.5 |
| | *tree-base* | 68.6 | 52.6 | 50.4 | 51.4 | 34.4 | 59.1 |
| *rooted* | *lstm* | 66.3 | 51.5 | 59.3 | 54.6 | 36.1 | 51.8 |
| *rooted* | *lstm+leninit* | 70.8 | 55.2 | 57.1 | 55.9 | 36.0 | 56.8 |
| *rooted* | *lstm+lenemb* | 71.0 | 55.2 | 56.7 | 55.7 | 35.9 | 57.8 |
| *multi-root+* | *lstm* | 60.8 | 50.6 | 60.5 | 54.1 | 33.7 | 47.9 |
| *multi-root+* | *lstm+leninit* | 71.7 | 56.0 | 57.8 | 56.7 | 36.3 | 57.8 |
| *multi-root+* | *lstm+lenemb* | 72.6 | 56.2 | 56.4 | 56.1 | 35.8 | 59.4 |

Table 2: Automatic evaluation. R-1, R-2, and R-L are ROUGE-1, ROUGE-2, and ROUGE-L score. R, P and F are recall, precision and F-measure.

tion 2.4). The properties of the dataset in relation to $\theta$ are shown in Table 1. $\theta$ is tuned for ROUGE-2 score on the development data. In Table 1, we show the tendency of the ROUGE-2 scores when *lstm+leninit* was trained on created *rooted* with each $\theta$. From Table 1, we chose 0.5 as $\theta$. All models are three stacked LSTM layers[7]. Words with frequency lower than five are regarded as unknown words. ADAM[8] was used as the optimization method. The desired length was set to the bytes of a compressed sentence randomly chosen from the five human-generated sentences. In the test step, beam-search was used (beam size: 20) and candidates exceeding the desired length were truncated.

**tree-base and prop-w-dpnd:** Existing methods for Japanese sentence compression are not based on the training on a large dataset. Therefore, the proposed method is compared with two methods, *tree-base*, (Filippova and Strube, 2008) and *prop-w-dpnd* (Harashima and Kurohashi, 2012), which are not based on supervised learning. *tree-base* is implemented as an integer linear programming problem that finds a subtree of a dependency tree. *prop-w-dpnd* is also implemented as an integer linear programming problem, but modified based on characteristics of the Japanese language. *prop-w-dpnd* allows the deletion inside the chunks.

### 4.1 Automatic evaluation

Table 2 shows the ROUGE score of each model. We used ROUGE-1, ROUGE-2, and ROUGE-L (R-1, R-2, and R-L) for evaluation. In addition, we also show the precision and F-measure of ROUGE-2 because the output length of each model is not same. Compression ratio (CR) is the ratio of the bytes of the compressed sentence to the bytes of the source sentence. Average CR of the test data is approximately 61.0%. We think we should focus on F-measure of ROUGE-2 because of the different CR of the models. The models trained on a large training dataset achieved higher F-measure than the unsupervised models. Moreover, F-measure of *lstm+leninit* and *lstm+lenemb*, which were trained on either *multi-root+* or *rooted*, is significantly better than *prop-w-dpnd* ($p < 0.001$). *lstm* achieved lower R-1 and R-2 than the other models trained on a large training dataset, probably because it tends to generate too short sentences, as indicated by low CR. It is also noteworthy that CRs of *lstm+leninit* and *lstm+lenemb* are mostly closer to the average CR of the test data than *lstm*. Furthermore, *lstm+leninit* and *lstm+lenemb* trained on *multi-root+* instead of *rooted* worked better in terms of F-measure. We consider it is because various types of deletion make the compression model more flexible, as indicated by closer CR of the model trained on *multi-root+* instead of *rooted* to the average CR of the test data.

### 4.2 Human evaluation

The difference between *lstm+leninit* and *lstm+lenemb* trained on *multi-root+* was investigated first. With *lstm+leninit*, 2 out of 100 sentences, chosen randomly, ended with a word that cannot be located at the end of a sentence. In contrast, with *lstm+lenemb*, 24 sentences ended with such words and therefore are ungrammatical, although *lenemb* has shown to be effective in abstractive sentence summarization (Kikuchi et al., 2016). This result suggests that *lstm+lenemb* is excessively affected by the desired length because *lenemb* receives the potential desired length at each time of decoding. In fact, 21 out of the 24 sentences are as long as the desired length.

Then, *lstm+leninit* trained on *multi-root+* was evaluated by crowdsourcing in comparison with the gold-standard and *tree-base*. Each crowd-

[7]Dimension of word embedding and hidden layer:256, dropout:20%

[8]$\alpha$:0.001, $\beta_1$:0.9, $\beta_2$:0.999, eps:1e-8, batchsize:180

| model | *read* | *info* |
|---|---|---|
| gold-standard | 4.22 | 3.76 |
| *lstm+leninit* | 3.99 | 3.09 |
| *prop-w-dpnd* | 3.55 | 2.81 |

Table 3: Human absolute evaluation

| model | *read* | *info* |
|---|---|---|
| *lstm+leninit* | 78 | 108 |
| *lstm* | 47 | 47 |
| *tie* | 125 | 95 |

| training data | *read* | *info* |
|---|---|---|
| *multi-root+* | 55 | 85 |
| *rooted* | 57 | 33 |
| *tie* | 138 | 132 |

Table 4: Human relative evaluation

sourcing worker reads each source sentence and a set of the compressed sentences, reordered randomly, and gives a score from 1 to 5 (where 5 is best) to each compressed sentence in terms of informativeness (*info*) and readability (*read*). As shown in Table 3, in terms of both *read* and *info*, *lstm+leninit* archived higher scores than *prop-w-dpnd*, and the difference is significant ($p < 0.001$).

Next, *lstm+leninit* was compared with *lstm* (both are trained on *multi-root+*), and *multi-root+* was compared with *rooted* (*lstm+leninit* was used as the model), by human relative evaluation. In this evaluation, each worker votes for one of *lstm+leninit*, *lstm*, or tie, and for one of *multi-root+*, *rooted*, or tie. 250 votes in total were received (50 sentences $\times$ 5 votes). The results are shown in Table 4. *lstm+leninit* is better than *lstm* in terms of *read*, which is not directly related to the output length, as well as of *info*. It is also clear that *multi-root+* achieves much higher *info* with a negligible reduction in *read* than *rooted*.

## 5 Conclusion

Filippova et al.'s method of creating training data for English sentence compression was modified to create a training dataset for Japanese sentence compression. The effectiveness of the created Japanese training dataset was verified by automatic and human evaluation. Our method can refine the created dataset by giving more flexibility in compression and achieved better informativeness with negligible reduction in readability. Furthermore, it has been shown that controlling the output length improves the performance of the sentence compression models.

## Acknowledgment

This work was supported by Google Research Award, JSPS KAKENHI Grant Number JP26280080, and JPMJPR1655.

## References

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*. pages 481–490.

James Clarke and Mirella Lapata. 2006. Models for sentence compression: A comparison across domains, training requirements and evaluation measures. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*. pages 377–384.

Katja Filippova, Enrique Alfonseca, Carlos A. Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 360–368.

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. pages 1481–1491.

Katja Filippova and Michael Strube. 2008. Dependency tree based sentence compression. In *Proceedings of the Fifth International Natural Language Generation Conference*. pages 25–32.

Jun Harashima and Sadao Kurohashi. 2012. Flexible Japanese sentence compression by relaxing unit constraints. In *Proceedings of COLING 2012*. pages 1097–1112.

Tsutomu Hirao, Jun Suzuki, and Hideki Isozaki. 2009. A syntax-free approach to japanese sentence compression. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. pages 826–833.

Chiori Hori and Sadaoki Furui. 2004. Speech summarization : An approach through word extraction and a method for evaluation (¡special section¿the 2002 ieice excellent paper award). *IEICE transactions on information and systems* pages 15–25.

Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. Controlling output length in neural encoder-decoders. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 1328–1338.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization - step one: Sentence compression. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*. pages 703–710.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*. pages 3104–3112.

Kapil Thadani and Kathleen McKeown. 2013. Sentence compression with joint structural inference. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*. pages 65–74.

Jenine Turner and Eugene Charniak. 2005. Supervised and unsupervised learning for sentence compression. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. pages 290–297.