Effectiveness of Syntactic Dependency Information for Higher-Order Syntactic Attention Network

Hidetaka Kamigaito†, Katsuhiko Hayashi††, Tsutomu Hirao†††, Masaaki Nagata††† and Manabu Okumura††††

Recently, as a replacement of syntactic tree-based approaches, such as tree-trimming, Long Short-Term Memory (LSTM)-based methods have been commonly used to compress sentences because LSTM can generate fluent compressed sentences. However, the performance of these methods degrades significantly while compressing long sentences because they do not explicitly handle long-distance dependencies between the words. To solve this problem, we proposed a higher-order syntactic attention network (HiSAN) that can handle higher-order dependency features as an attention distribution on LSTM hidden states. Furthermore, to avoid the influence of incorrect parse results, we trained HiSAN by maximizing the probability of a correct output together with the attention distribution. Experiments on the Google sentence compression dataset show that our method improved the performance from baselines in terms of F$_1$ as well as ROUGE-1, -2, and -L scores. In subjective evaluations, HiSAN outperformed baseline methods in both readability and informativeness. Besides, in this study, we additionally investigated the performance of HiSAN after training it without any syntactic dependency tree information. The results of our investigation show that HiSAN can compress sentences without relying on any syntactic dependency information while maintaining accurate compression rates, and also shows the effectiveness of syntactic dependency information in compressing long sentences with higher F$_1$ scores.

Key Words: Sentence Compression, Sequence-to-Sequence, Syntactic Dependency

1 Introduction

Syntax information plays an important role in sentence compression, a task that involves compressing long sentences into short and concise ones by deleting words. To generate grammatically correct compressed sentences in this task, several researchers (Jing 2000; Knight and Marcu

---

† NTT Communication Science Laboratories, NTT Corporation. Currently, he is working in Institute of Innovative Research, Tokyo Institute of Technology.

†† NTT Communication Science Laboratories, NTT Corporation. Currently, he is working at Gunma University.

††† NTT Communication Science Laboratories, NTT Corporation

†††† Institute of Innovative Research, Tokyo Institute of Technology
2000; Berg-Kirkpatrick et al. 2011; Filippova and Altun 2013) have adopted tree trimming methods. Although Fillipova and Altun (2013) reported the best results in this task, parsing errors greatly degrade the performance of the tree trimming methods. Recently, Filippova et al. (2015) proposed an LSTM sequence-to-sequence (Seq2Seq)-based sentence compression method that can generate fluent sentences without utilizing any syntactic features. Therefore, Seq2Seq-based sentence compression is a promising alternative to tree trimming.

However, as reported in a machine translation task (Cho et al. 2014; Pouget-Abadie et al. 2014; Koehn and Knowles 2017), the longer the input sentences, the worse the Seq2Seq performance. We also observed this problem in sentence compression tasks. As shown in Fig. 1, the performance of Seq2Seq degrades while compressing long sentences. In particular, the performance significantly drops if the sentence length exceeds 26 words. This is an important problem, because sentences longer than the average sentence length (27.04 words) account for 42\% of the Google sentence compression dataset.

As shown in Fig. 2, long sentences have deep dependency trees, with a large distance between the root node and the words at the leaf nodes. Therefore, focusing on improving the compression performance for sentences with such deep dependency trees can help in compressing long sentences.

Besides, as shown in Fig. 3, there is a tendency of longer compressed sentences for longer inputs in the gold compressed sentences. Furthermore, as shown in Fig. 4, similar to the relationship between input sentences and their dependency trees, the average depth of the gold compressed sentences increases for long compressed sentences. These observations suggest that to improve the compression performance, we need to be careful about deep dependency trees on both the

---

1 We used an LSTM-based sentence compression method (Filippova et al. 2015) during evaluation, as described in Section 4.1.

2 We treated the maximum distance from a root node to the leaf node as the dependency tree depth.
input and output sides.

To deal with sentences that have deep dependency trees, we focused on the chains of dependency relationships. Fig. 5 shows an example of a compressed sentence with its dependency tree. The topic of this sentence is import agreement related to electricity. Thus, to generate an informative compression, the compressed sentence must retain the country name. In this example, the compressed sentence should retain the phrase “from Kyrgyz Republic and Tajikistan”. Thus, the compressed sentence must also retain the dependency chain comprising the words “import”, “resolution”, and “signed” because the phrase is a child of this chain. By considering such higher-order dependency chains, the system can implement informative compression. The example in Fig. 5 demonstrates that tracking a higher-order dependency chain for each word helps in compressing long sentences. This paper refers to such dependency relationships by the expression “$d$-length dependency chains”.

To handle a $d$-length dependency chain for sentence compression with LSTM, we had earlier proposed a technique called the higher-order syntactic attention network (HiSAN) (Kamigaito et al. 2018). HiSAN computes the deletion probability for a given word based on the $d$-length dependency chain starting from that word. The $d$-length dependency chain is represented as an attention distribution, learned from automatic parse trees. The attention distribution of HiSAN is calculated from both the input and the compressed sentences. To alleviate the influence of parse errors in automatic parse trees, we learned the attention distribution together with the

---

**Fig. 3** Compressed length for each input sentence length in the gold compression. The dot size represents the number of each length.

**Fig. 4** Average tree depths for each compression length in the gold compression.

**Fig. 5** An example compressed sentence and its dependency tree. The words colored with gray represent deleted words.

323
deletion probability.

Evaluation results on the Google sentence compression dataset (Filippova and Altun 2013) demonstrated that HiSAN improved the sentence compression performance from the baseline in terms of $F_1$, ROUGE-1, -2, and -L scores. In particular, HiSAN attained remarkable compression performance with long sentences. HiSAN outperformed the baseline methods even in human evaluations.

Although the attention module of HiSAN can learn context information without relying on external syntactic dependency information, in our preliminary study (Kamigaito et al. 2018), we had evaluated HiSAN in a setting where the syntactic dependency information was utilized in a supervised manner only. Thus, it is still uncertain which network structure or syntactic dependency information contributed to the improvement in the performance. Identifying this point will help to construct a model for more accurate sentence compression. In this study, we additionally investigated the performance of HiSAN, which was trained without any syntactic dependency tree information. Our investigation indicates that HiSAN can compress sentences with high accuracy rates without relying on any syntactic dependency information. It also shows the effectiveness of syntactic dependency information in compressing long sentences with higher $F_1$ scores while maintaining accurate compression rates.

The remainder of this paper is structured as follows: Section 2 introduces the baseline Seq2Seq method which is used for constructing HiSAN; Section 3 describes the concept of $d$-length dependency chain and HiSAN network structures; Section 4 describes the comparison between baselines and the variants of HiSAN using automatic and human evaluations; Section 5 analyzes the evaluation results with real generated sentences and attention distributions; Section 6 describes the related work used for this paper; Section 7 concludes the paper.

2 Baseline Seq2Seq Method

Sentence compression can be regarded as a tagging task on a given sequence of input tokens $\mathbf{x} = (x_0, ..., x_n)$, where $x_0$ represents the root node,\(^3\) and the system assigns one out of three specific label types (“keep”, “delete”, or “end of sentence”) as an output label $y_t$ to each input token $x_t$ ($1 \leq t \leq n$).

LSTM-based approaches for sentence compression are mostly based on either a bidirectional-LSTM (bi-LSTM)-based tagging method (Tagger) (Klerke et al. 2016; Wang et al. 2017; Chen

\(^3\) In this paper, we treat $x_0$ as the root node of a dependency tree.
and Pan 2017) or Seq2Seq (Filippova et al. 2015; Tran et al. 2016). Tagger independently predicts the labels in a point estimation manner, whereas Seq2Seq predicts them by considering previously predicted labels. Because Seq2Seq is more expressive than Tagger, we developed HiSAN on the baseline Seq2Seq model.

Our baseline Seq2Seq is a variant of the model proposed by Filippova et al. (2015), wherein we added the bi-LSTM, an input feeding approach (Vinyals et al. 2015; Luong et al. 2015), and a monotonic hard attention method (Yao and Zweig 2015; Tran et al. 2016). As described in the evaluations section, this baseline achieved comparable or even better scores than those reported for the state-of-the-art method in Filippova et al. (2015). Our baseline Seq2Seq model comprised the embedding, encoder, decoder, and output layers.

The embedding layer converts the input tokens \( x \) into embeddings \( e \). As reported in Wang et al. (2017), syntactic features are important to learn a generalizable embedding for sentence compression. Following their results, we also introduced syntactic features into the embedding layer. Specifically, we combined surface token embedding \( w_i \), Part-Of-Speech (POS) tag embedding \( p_i \), and dependency relation label embedding \( r_i \) into a single vector as follows:

\[
e_i = [w_i, p_i, r_i],
\]

where \([\ ]\) represents the vector concatenation, and \( e_i \) is an embedding of token \( x_i \).

The encoder layer converts the embedding \( e \) into a sequence of hidden states \( h = (h_0, ..., h_n) \) using a stacked bi-LSTM as follows:

\[
h_i = \left[ \overset{\rightarrow}{h}_i, \overset{\leftarrow}{h}_i \right] \tag{2}
\]

\[
\overset{\rightarrow}{h}_i = LSTM_{\theta}(\overset{\rightarrow}{h}_{i-1}, e_i) \tag{3}
\]

\[
\overset{\leftarrow}{h}_i = LSTM_{\theta}(\overset{\leftarrow}{h}_{i+1}, e_i), \tag{4}
\]

where \( LSTM_{\theta} \) and \( LSTM_{\theta} \) represent the forward and backward LSTM functions, respectively. The final state of the backward LSTM \( \overset{\leftarrow}{h}_0 \) is inherited by the decoder as its initial state.

In the decoder layer, the concatenation of a three-bit one-hot vector, determined by a previously predicted label \( y_{t-1} \), the previous final hidden state \( d_{t-1} \) (explained later), and the input embedding of \( x_t \), is encoded into the decoder hidden state \( s_t \) using stacked forward LSTMs.

Contrary to the original softmax attention method, we can deterministically focus on one encoder hidden state \( h_t \) (Yao and Zweig 2015) to predict \( y_t \) in the sentence compression task.
In the output layer, label probability is calculated as follows:

\[
P(y_t | y_{<t}, x) = \text{softmax}(W_o \cdot d_t) \cdot \delta_{y_t},
\]

(5)

\[
d_t = [h_t, \overrightarrow{s}_t],
\]

(6)

where \(W_o\) is the weight matrix of the softmax layer and \(\delta_{y_t}\) is a binary vector where the \(y_t\)-th element is set to 1 and the other elements to 0.

3 Higher-order Syntactic Attention Network

The key component of HiSAN is its attention module. Unlike the baseline Seq2Seq, HiSAN employs a packed \(d\)-length dependency chain as distributions in the attention module. Section 3.1 explains the packed \(d\)-length dependency chain, Section 3.2 describes the network structure of our attention module, and Section 3.3 explains the learning method of HiSAN.

3.1 Packed \(d\)-length Dependency Chain

The probability of a packed \(d\)-length dependency chain is obtained from a dependency graph, which is an edge-factored dependency score matrix (Hashimoto and Tsuruoka 2017; Zhang et al. 2017). First, we explain the dependency graph. Fig. 6 (a) shows an example of the dependency graph to explain a parent attention module. HiSAN represents the dependency graph as an attention distribution generated by the attention module. A probability for each dependency edge is obtained from the attention distribution.

Fig. 6 (b) explains a recursive attention module by showing an example of the packed \(d\)-length dependency chain. With our recursive attention module, the probability of a packed \(d\)-length dependency chain is computed as the sum of probabilities of each path yielded by recursively tracking from a word to its \(d\)-th ancestor. The probability of each path is calculated as the product of the probabilities of the tracked edges. The packed \(d\)-length dependency chains can compactly represent the probability of the chain. A similar effect can be observed while considering multiple dependency trees and their \(d\)-th dependency parents. Thus, this representation can alleviate the incorrect parse results, because of the nondependence on a single dependency tree. This is the advantage of using dependency graphs.

---

4 This is because the output and input lengths are the same, and each \(x_t\) can be assigned to a \(y_t\) in a one-to-one correspondence.
3.2 Network Architecture

Fig. 7 shows the HiSAN prediction process. The figure depicts the prediction of output label $y_7$ from the input sentence using HiSAN. The prediction process is as follows.

1) **Parent Attention** module calculates $P_{\text{parent}}(x_j|x_t, x)$, the probability of $x_j$ being the parent of $x_t$, using $h_j$ and $h_t$. This probability is calculated for all pairs of $x_j, x_t$. The arc in Fig. 7 shows the most probable dependency parent for each child token.

2) **Recursive Attention** module calculates $\alpha_{d,t,j}$, the probability of $x_j$ being the $d$-th order parent ($d$ denotes the chain length) of $x_t$, by recursively using $P_{\text{parent}}(x_j|x_t, x)$. $\alpha_{d,t,j}$ is also treated as an attention distribution, and used to calculate $\gamma_{d,t}$, the weighted sum of $h$ for each length $d$. For example, a three-length dependency chain of word $x_7$ with the highest probability is $x_6$-$x_5$-$x_2$. The encoder hidden states $h_6$, $h_5$, and $h_2$, which correspond to the dependency chain, are weighted by calculated parent probabilities $\alpha_{1,7,6}$, $\alpha_{2,7,5}$, and $\alpha_{3,7,2}$, respectively, and then fed to the selective attention module.

3) **Selective Attention** module calculates weight $\beta_{d,t}$ from its length, $d \in d$, for each $\gamma_{d,t}$, $d$ represents a group of chain lengths. $\beta_{d,t}$ is calculated from the encoder and decoder hidden states. Each $\beta_{d,t} \cdot \gamma_{d,t}$ is then summed to $\Omega_t$, the output of the selective attention module.
module.

(4) Finally, the calculated $\Omega_t$ is concatenated and input into the output layer. Each module is explained in detail in the following subsections.

### 3.2.1 Parent Attention Module

Zhang et al. (2017) formalized dependency parsing as the problem of independently selecting the parent of each word in a sentence. They produced a distribution over the possible parents of each child word by using the attention layer on bi-LSTM hidden layers.

In a dependency tree, a parent has more than one child. Under this constraint, dependency parsing is represented as follows. Given a sentence $S = (x_0, x_1, ..., x_n)$, the parent of $x_j$ is selected from $S \setminus x_i$ for each token $S \setminus x_0$. Notably, $x_0$ denotes the root node. The probability of token $x_j$ being the parent of token $x_t$ in sentence $\mathbf{x}$ is calculated as follows:

$$P_{parent}(x_j|x_t, \mathbf{x}) = \text{softmax}(g(h_j', h_t)) \cdot \delta_{x_j}, \quad (7)$$

$$g(h_j', h_t) = v^T_a \cdot \text{tanh}(U_a h_j' + W_a h_t), \quad (8)$$
where $j'$ indicates all the possible indices of a word in $S$, $v_a$ is a weight vector, and $U_a$ and $W_a$ are the weight matrices of $g$.

Different from the attention-based dependency parser, $P_{parent}(x_j|x_t, x)$ is jointly learned with output label probability $P(y \mid x)$ in the training phase. The training details are explained in Section 3.3.

### 3.2.2 Recursive Attention Module

The recursive attention module recursively calculates $\alpha_{d,t,j}$, the probability of $x_j$ being the $d$-th order parent of $x_t$, as follows:

$$
\alpha_{d,t,j} = \begin{cases} 
\sum_{k=1}^{n} \alpha_{d-1,t,k} \cdot \alpha_{1,k,j} & (d > 1) \\
\cdot P_{parent}(x_j|x_t, x) & (d = 1)
\end{cases}
$$

(9)

Furthermore, in a dependency parse tree, the root should not have any parent, and a token should not depend on itself. To satisfy these rules, we impose the following constraints on $\alpha_{1,t,j}$:

$$
\alpha_{1,t,j} = \begin{cases} 
1 & (t = 0 \land j = 0) \\
0 & (t = 0 \land j > 0) \\
0 & (t \neq 0 \land t = j)
\end{cases}
$$

(10)

The first and second lines of Eq. (10) represent the case where the parent of root is also root. These constraints imply that root does not have any parent. The third line of Eq. (10) prevents a token from depending on itself. Because the first line of Eq. (9) is a matrix multiplication, Eq. (9) can be efficiently computed on a CPU and GPU.5

By recursively using the single attention distribution, it is no longer necessary to prepare additional attention distributions for each order when computing the probability of higher order parents. Furthermore, owing to the nonnecessity of learning multiple attention distributions, using hyper-parameters for adjusting the weight of each distribution in training is not required. Finally, this method can prevent the problem of sparse higher-order dependency relations in the training dataset.

The $\alpha_{d,t,j}$ calculated above is used to weight the bi-LSTM hidden layer $\mathbf{h}$ as follows:

$$
\gamma_{d,t} = \sum_{k=j}^{n} \alpha_{d,t,j} \cdot h_j.
$$

(11)

5 During training, HiSAN with 1- and 3-length dependency chains consumed 25 and 26 min, respectively, per epoch on an Intel Xeon E5-2697 v3 2.60 GHz.
Notably, $\gamma_{d,t}$ is inherited by the selective attention module, as explained in the next section.

### 3.2.3 Selective Attention Module

To select suitable dependency orders of the input sentence, the selective attention module weights and sums the hidden states $\gamma_{d,t}$ to $\Omega_t$ by using weighting parameter $\beta_{d,t}$, according to the current context as follows:

\[
\beta_{d,t} = \text{softmax}(W_c c_t) \cdot \delta_d, \\
\Omega_t = \sum_{d \in \{0, d\}} \beta_{d,t} \cdot \gamma_{d,t},
\]

where $W_c$ is the weight matrix of the softmax layer, $d$ is a group of chain lengths, $c_t$ is a vector representing the current context, $\gamma_{0,t}$ is a zero-vector, and $\beta_{0,t}$ indicates the weight when the method does not use the dependency features. Context vector $c_t$ is calculated as $c_t = [\overrightarrow{h_0}, \overrightarrow{h_n}, \overrightarrow{s_t}]$ using the current decoder hidden state $\overrightarrow{s_t}$.

The calculated $\Omega_t$ is concatenated and input into the output layer. In detail, $d_t$ in Eq. (5) is replaced by concatenated vector $d'_t = [h_t, \Omega_t, \overrightarrow{s_t}]$; furthermore, instead of $d_t$, $d'_t$ is also fed to the input of the decoder LSTM at $t+1$. It should be noted that considering a high order dependency structure helps to deal with long input sentences. However, when the number of order types increases, estimating their importance is difficult using selective attention. Thus, an appropriate order combination is required.

### 3.3 Objective Function

To alleviate the influence of parse errors, we jointly update the first-order attention distribution $\alpha_{1,t,k}$ and label probability $P(y|x)$ (Kamigaito et al. 2017). The first-order attention distribution is learned by dependency parse trees. If $a_{t,j} = 1$ is an edge between parent and child words $w_j$ and $w_t$, respectively, on a dependency tree ($a_{t,j} = 0$ denotes that $w_j$ is not a parent of $w_t$), the objective function of our method can be defined as:

\[
-\log P(y|x) - \lambda \cdot \sum_{j=1}^{n} \sum_{t=1}^{n} a_{t,j} \cdot \log \alpha_{1,t,j},
\]

where $\lambda$ is a hyper-parameter that controls the importance of the output labels and parse trees in the training dataset. To investigate the effectiveness of the information from syntactic dependency trees, we set $\lambda = 1.0$ for with syntax (w/ syn) setting and $\lambda = 0.0$ for without syntax (w/o syn) setting.
4 Evaluation

4.1 Evaluation Settings

4.1.1 Dataset

For evaluation we used the Google sentence compression dataset (Filippova and Altun 2013). This dataset contains information regarding the compression labels, part-of-speech (POS) tags, dependency parents, and dependency relation labels of each sentence. We used the first and last 1,000 sentences of comp-data.eval.json as our test and development datasets, respectively. Notably, our test dataset is compatible with that used in previous studies (Filippova et al. 2015; Tran et al. 2016; Klerke et al. 2016; Wang et al. 2017). In this study, we trained the baselines and HiSAN on all the sentences present in the file sent-comp.train*.json (200,000 sentences in total). In our experiments, we replaced rare words that appear fewer than 10 times in our training dataset with a special token \langle UNK \rangle. The resultant filtered input vocabulary size was 23,168.

4.1.2 Comparison Methods

For a fair comparison with HiSAN, we used the input features described in Eq. (1) in the following baseline methods:

- **Tagger**: This method regards sentence compression as a tagging task based on bi-LSTM (Klerke et al. 2016; Wang et al. 2017).

- **Tagger+ILP**: This is an extension of **Tagger** that integrates integer linear programming (ILP)-based dependency tree trimming (Wang et al. 2017). Here, we set the positive parameter \( \lambda \) to 0.2.

- **Bi-LSTM**: This method, proposed by Filippova et al. (2015), regards sentence compression as a Seq2Seq translation task. For a fair comparison, we replaced their one-directional LSTM with the more expressive bi-LSTM in the encoder part. The initial state of the decoder was set to the sum of the final states of the forward and backward LSTMs.

- **Bi-LSTM-Dep**: This is an extension of Bi-LSTM that exploits the features obtained from a dependency tree (named LSTM-PAR-PRES in Filippova et al. (2015)). Following

---

6 https://github.com/google-research-datasets/sentence-compression

7 Notably, Filippova et al. (2015) used 2,000,000 sentences for training their method, but the datasets are not publicly available.

8 Please note that the large training dataset lacks periods at the end of compressed sentences. To unify the form of compressed sentences in small and large settings, we added periods to the end of compressed sentences in the large training dataset.

331
their work, we fed the word embedding and predicted label of a dependency parent word to the current decoder input of Bi-LSTM.

- **Base**: This is our baseline Seq2Seq method described in Section 2.
- **Attn**: This method extends the softmax-based attention method (Luong et al. 2015). We replaced $h_t$ in Eq. (6) with the weighted sum calculated with the commonly used concat attention (Luong et al. 2015).
- **HiSAN-Dep**: This variant of HiSAN utilizes the pipeline approach. We fixed $\alpha^{1,j,t}$ to 1.0 if $x_j$ was a parent of $x_t$ in the input dependency parse tree, otherwise 0.0. In this baseline, $d = \{1\}$ was used.

Our proposed methods are as follows:

- **HiSAN-Dep**: Different from HiSAN-Dep in the baseline setting, we used higher-order dependencies $d = \{1, 2\}, \ d = \{1, 2, 3\}, \ d = \{1, 2, 4\}$, and $d = \{1, 2, 3, 4\}$ in this setting.
- **HiSAN w/ syn**: As explained in Section 3.3, this setting considers syntactic dependency information for training HiSAN by setting $\lambda$ to 1.0.
- **HiSAN w/o syn**: Different from HiSAN w/ syn, this setting does not consider syntactic dependency information for training HiSAN. $\lambda$ is set to 0.0 for excluding any influence of syntactic dependency information.

We list the number of weight parameters of the comparison methods in Table 1.

### Table 1

| Method          | Weight Parameters | Dependency Order | Weight Parameters | Dependency Order |
|-----------------|-------------------|------------------|-------------------|------------------|
| Tagger          | 2,880,004         |                  | HiSAN-Dep        | 3,195,807        |
| Tagger+ILP      | 2,880,004         |                  | HiSAN w/ syn     | 3,196,108        |
| Bi-LSTM         | 2,963,304         |                  | HiSAN-Dep        | 3,196,409        |
| Bi-LSTM-Dep     | 3,036,504         |                  | HiSAN w/ syn     | 3,196,108        |
| Attn            | 3,104,104         |                  | HiSAN-Dep        | 3,195,506        |
| Base            | 3,074,004         |                  | HiSAN w/ syn     | 3,196,108        |
| HiSAN-Dep       | (d = \{1\})      |                  | HiSAN w/ syn     | 3,196,409        |
| HiSAN w/ syn    | (d = \{1\})      |                  | HiSAN w/o syn    | 3,195,506        |
| HiSAN w/o syn   | (d = \{1\})      |                  | HiSAN w/ syn     | 3,196,108        |

**4.1.3 Training Details**

Following the previous work (Wang et al. 2017), the dimensions of the word embeddings, LSTM layers, and attention layer were set to 100. For the Tagger-style and Seq2Seq-style methods, the depth of the LSTM layer was set to three and two, respectively. In this setting, all methods have a total of six LSTM-layers. The dimensions of POS and dependency-relation label embeddings were set to 40. All parameters were initialized as per the method described by Glorot and Bengio (2010). For all methods, we applied Dropout (Srivastava et al. 2014) to the input of
Kamigaito et al. Effectiveness of Syntactic Dependency Information for HiSAN

LSTM layers. All the dropout rates were set to 0.3.

During training, the learning rate was tuned with Adam (Kingma and Ba 2014). The initial learning rate was set to 0.001. The maximum number of training epochs was set to 30. All the gradients were averaged in each mini-batch. The maximum mini-batch size was set to 16. The order of mini-batches was shuffled at the end of each training epoch. The clipping threshold of the gradient was set to 5.0. We selected trained models with early stopping for maximizing the per-sentence accuracy (i.e., how many compressions could be fully reproduced) of the development dataset.

To obtain a compressed sentence, we used greedy decoding instead of beam search decoding, because the latter did not demonstrate better performance in the development dataset. This may relate to the small search space of this task, because the number of labels is only 3. All the methods were implemented in C++ on Dynet (Neubig et al. 2017).

4.2 Automatic Evaluation

For automatic evaluation, we used token-level F$_1$-measure (F$_1$) as well as the recall scores of ROUGE-1, -2, and -L (Lin and Och 2004). We used the ROUGE-1.5.5 script for calculating the ROUGE scores. When calculating the ROUGE-1 score, we excluded stop words by using options “-n 1 -m -d -s -a” to consider the informativeness of the compressed sentences. When calculating the ROUGE-2 and ROUGE-L scores, we used options “-n 2 -m -d -a” to avoid meaningless collocations. For a fair comparison while calculating ROUGE scores, if a system output exceeded the reference summary byte length, we truncated the exceeding tokens.

We used $\Delta C = \text{system compression ratio} - \text{gold compression ratio}$ to evaluate the closeness of the compression ratio of system outputs to that of gold compressed sentences. Notably, we calculated $\Delta C$ in this study after rounding the compression ratios for each method. We used micro-averages for $F_1$-measure and compression ratio, and macro-averages for the ROUGE scores, respectively.

To verify the superiority of our methods on long sentences, we additionally reported the scores on sentences longer than the average sentence length (= 27.04) in the test set.

All the results are reported as the average scores and their standard deviation of five trials. In each trial, different random choices were used to generate the initial values of the embeddings and the order of mini-batch processing.

Table 2 and 3 present the results with the setting for all sentences (ALL) and long sentences (LONG), respectively. The HiSANs outperformed the other methods. In particular, HiSAN w/ syn ($d = \{1, 2, 4\}$) achieved the best score on $F_1$, ROUGE-2, ROUGE-L, and $\Delta C$ in the ALL
setting. In addition, the improvements with HiSAN w/ syn \((d = \{1, 2, 4\})\) in \(F_1\) and ROUGE scores from the baselines methods in the LONG setting are larger than those in the ALL setting. HiSAN w/o syn \((d = \{1, 2, 3, 4\})\) achieved comparable scores with HiSAN w/ syn \((d = \{1, 2, 4\})\) in the ALL setting. In the LONG setting, HiSAN w/o syn \((d = \{1, 2, 3, 4\})\) achieved higher ROUGE scores compared with HiSAN w/ syn. This result indicates that our recursive attention module can retain important words in the compressed sentence without considering syntactic dependency information. However, in terms of \(F_1\), HiSAN w/o syn scores were lower than those of HiSAN w/ syn \((d = \{1, 2, 4\})\) in both ALL and LONG settings, which suggests that considering explicit syntactic dependency information can reduce redundant words during sentence compression. From these results, we can conclude that \(d\)-length dependency chains

![Table 2](image)

Results of automatic evaluation for all sentences in the test dataset. ± denotes the standard deviation for each score. CR denotes the compression ratio. \(d\) represents the groups of \(d\)-length dependency chains. * indicates the best scoring model among similar methods with different \(d\) in the development dataset. Bold values indicate the best scores.
Table 3  Results of automatic evaluation for long sentences (longer than the average length, 27.04) in the test dataset. The notation used is the same as in Table 2.

|                | F₁     | ROUGE I  | ROUGE 2  | ROUGE L  | ΔC     | CR    |
|----------------|--------|----------|----------|----------|--------|-------|
| Gold           |        | -        | -        | -        | -      | 31.4  |
| Tagger         | 80.4 ± 0.4 | 73.9 ± 0.8 | 65.5 ± 0.9 | 73.8 ± 1.0 | -2.8 ± 0.9 | 28.6 ± 0.9 |
| Tagger+ILP     | 75.4 ± 0.3 | 67.7 ± 0.7 | 57.7 ± 0.8 | 68.5 ± 0.7 | -3.6 ± 0.6 | 27.8 ± 0.6 |
| Bi-LSTM        | 78.9 ± 0.7 | 73.2 ± 1.3 | 66.3 ± 1.3 | 73.5 ± 1.4 | -2.1 ± 0.4 | 29.3 ± 0.4 |
| Bi-LSTM-Dep    | 79.6 ± 0.4 | 74.1 ± 0.5 | 67.4 ± 0.5 | 74.5 ± 0.5 | -1.9 ± 0.7 | 29.5 ± 0.7 |
| Attn           | 79.8 ± 0.5 | 74.4 ± 0.5 | 67.8 ± 0.7 | 74.8 ± 0.5 | -2.2 ± 0.4 | 29.2 ± 0.4 |
| Base           | 80.1 ± 0.8 | 74.9 ± 1.1 | 68.0 ± 1.3 | 74.9 ± 1.1 | -2.3 ± 0.6 | 29.1 ± 0.6 |
| HiSAN-Dep      |        |          |          |          |        |       |
| (d = {1})      | 80.1 ± 0.8 | 74.8 ± 0.7 | 68.1 ± 1.0 | 75.0 ± 0.9 | -1.9 ± 0.6 | 29.5 ± 0.6 |
| (d = {1, 2})   | 80.5 ± 0.9 | 75.3 ± 0.9 | 68.6 ± 1.3 | 75.5 ± 1.0 | -2.2 ± 0.4 | 29.2 ± 0.4 |
| (d = {1, 2, 3})| 80.5 ± 0.8 | 74.7 ± 1.1 | 68.1 ± 1.3 | 75.0 ± 1.1 | -2.3 ± 0.4 | 29.1 ± 0.4 |
| (d = {1, 2, 4})| 80.4 ± 0.3 | 75.0 ± 0.5 | 68.4 ± 0.6 | 75.3 ± 0.6 | -2.2 ± 0.4 | 29.2 ± 0.4 |
| (d = {1, 2, 3, 4}) | 80.1 ± 0.4 | 74.5 ± 0.3 | 67.7 ± 0.5 | 74.9 ± 0.5 | -2.1 ± 0.7 | 29.3 ± 0.7 |
| HiSAN w/ syn   |        |          |          |          |        |       |
| (d = {1, 2, 3})| 80.7 ± 0.4 | 75.6 ± 0.4 | 69.1 ± 0.5 | 75.9 ± 0.5 | -2.1 ± 0.5 | 29.3 ± 0.5 |
| (d = {1, 2, 4})| 80.9 ± 0.3 | 75.4 ± 0.5 | 69.0 ± 0.7 | 75.8 ± 0.7 | -1.8 ± 0.7 | 29.6 ± 0.7 |
| (d = {1, 2, 3, 4}) | 80.6 ± 1.1 | 75.2 ± 0.2 | 68.5 ± 1.9 | 75.5 ± 1.9 | -2.3 ± 0.6 | 29.1 ± 0.6 |
| HiSAN w/o syn  |        |          |          |          |        |       |
| (d = {1, 2, 3})| 80.0 ± 0.7 | 74.9 ± 0.2 | 68.0 ± 0.4 | 75.0 ± 0.2 | -2.3 ± 0.7 | 29.1 ± 0.7 |
| (d = {1, 2})   | 80.3 ± 0.4 | 74.8 ± 0.2 | 68.1 ± 0.2 | 75.1 ± 0.2 | -2.3 ± 0.3 | 29.1 ± 0.3 |
| (d = {1, 2, 4})| 80.5 ± 0.6 | 74.9 ± 0.8 | 68.3 ± 0.9 | 75.2 ± 0.9 | -2.2 ± 0.8 | 29.2 ± 0.8 |
| (d = {1, 2, 3, 4}) | 80.5 ± 0.3 | 76.0 ± 0.6 | 69.1 ± 0.6 | 76.0 ± 0.5 | -1.8 ± 0.3 | 29.6 ± 0.3 |

Table 3  Results of automatic evaluation for long sentences (longer than the average length, 27.04) in the test dataset. The notation used is the same as in Table 2.

are effective for sentence compression, especially in the case of longer than average sentences. HiSAN (d = {1}) outperformed HiSAN-Dep in F₁ scores in both ALL and LONG settings. This result shows the effectiveness of joint learning the dependency parse tree and output labels.

### 4.3 Human Evaluation

For human evaluation, we compared the baselines with our method, which achieved the highest F₁ score in the automatic evaluations. Sentences listing only a large number of multiple entities or containing long quotations are unsuitable for manual evaluation owing to various possible interpretations. Thus, we first removed such sentences from the test dataset. Then, we selected the first 100 sentences longer than the average sentence length (= 27.04) of the filtered test set in the order of appearance for human evaluation. It should be noted that when removing such
sentences, we identified them by the following rules: the sentence includes more than ten commas; the sentence includes a quotation symbol (“). Notably, we determined the number of commas by verifying that the human evaluation dataset did not contain any removable sentences. We show the statistics of the selected sentences in Fig. 8 and 9.

Similar to Filippova et al. (2015), a compressed sentence was rated by five annotators who were asked to select a rating on a five-point Likert scale, ranging from one to five for readability (Read) and for informativeness (Info). We report the average of these scores from the five raters. To investigate the differences between the methods, we also compared the baseline methods and HiSAN by using those sentences that displayed different compressions with each method.

Table 4 shows the results. HiSAN w/ syn ($d = \{1, 2, 4\}$) achieved better results than the baselines in terms of both readability and informativeness. The results agree with those obtained from the automatic evaluation. From the results of the sentences with different compressions between the baseline and HiSAN w/ syn ($d = \{1, 2, 4\}$), we can clearly observe the improvement attained by HiSAN w/ syn ($d = \{1, 2, 4\}$) in informativeness.

![Fig. 8](image1.png) The frequencies for each sentence length.  
![Fig. 9](image2.png) The frequencies for each dependency depth.

|     | Read | Info | CR  |
|-----|------|------|-----|
| **All** Tagger | 4.54 | 3.41 | 30.9 |
| (100) Base | 4.64 | 3.45 | 31.1 |
| HiSAN w/ syn ($d = \{1, 2, 4\}$) | 4.68 | 3.52 | 31.6 |
| **Diff** Base | 4.79 | 3.46 | 29.4 |
| (41) HiSAN w/ syn ($d = \{1, 2, 4\}$) | 4.89 | 3.64 | 30.6 |

Table 4 Results of human evaluation. **All** denotes the results for all sentences in the test set, and **Diff** represents the results for the sentences with different compressions for each method. Parentheses () denote the sentence size. The average gold compression ratio for input sentences in **All** and **Diff** were 32.1 and 31.5, respectively. Other notations are similar to those in Table 2.
5 Analysis

Fig. 10 presents the $F_1$ scores of each method for each sentence length. Notably, the HiSANs presented in this figure is the model that achieved the best $F_1$ scores in the validation dataset. In results for sentence lengths longer than 45, we can obviously observe that the syntactic information surpasses the performance degradation of Seq2Seq models. Tagger is also effective for such sentences because it does not have a decoder to memorize previously predicted labels for a correct prediction, and thus, it can deal with long sentences. However, the entire compression performance of Tagger is lower than Seq2Seq-based methods. In sentences with 45 words or less, HiSANs achieved high scores regardless of syntactic information. These results show that the network structure of HiSAN itself contributes to the improvement of $F_1$ scores.

Table 5 shows the examples of source sentences and their compressed variants generated by the baseline and HiSANs. We chose the model with the highest $F_1$ in the test set after five trials. For both examples, the compressed sentence output achieved by Base is grammatically correct. However, the informativeness is inferior to that attained by HiSAN w/ syn ($d = \{1, 2, 4\}$). The compressed sentence output by HiSAN-Dep in the second example lacks both readability and informativeness. Because HiSAN-Dep employs features obtained from the dependency tree in the pipeline procedure, it is influenced by the parsed tree. Based on this information, we checked the actual parsed tree and observed that the parent of “US” is wrongly predicted as “whistler”, instead of “Manning”. Zewei et al. (2020) revealed that Seq2Seq produces output related to the words that are indicated by attention. In the case of HiSAN-Dep, we can observe that the content of the wrongly stored parts in the compression is actually related to “whistler”. From this observation, we believe that the compression failure in this case is caused by the incorrect parse result. As reported in papers (Klerke et al. 2016; Wang et al. 2017), the $F_1$ scores of
Pakistan signed a resolution on Monday to import 1,300 MW of electricity from Kyrgyz Republic and Tajikistan to overcome power shortage in summer season, said an official press release.

| Input | Pakistan signed a resolution to import 1,300 MW of electricity from Kyrgyz Republic and Tajikistan. |
|-------|---------------------------------------------------------------------------------------------------|
| Gold  | Pakistan signed a resolution to import 1,300 MW of electricity from Kyrgyz Republic and Tajikistan. |
| Tagger| Pakistan signed a resolution to import 1,300 MW of electricity from Kyrgyz Republic and Tajikistan to overcome shortage. |
| Tagger-ILP | Pakistan signed resolution to import MW said. |
| Base  | Pakistan signed a resolution to import 1,300 MW of electricity. |
| HiSAN-Dep ($d = \{1\}$) | Pakistan signed a resolution to import 1,300 MW of electricity. |
| HiSAN w/ syn ($d = \{1, 2, 4\}$) | Pakistan signed a resolution to import 1,300 MW of electricity from Kyrgyz Republic and Tajikistan. |
| HiSAN w/o syn ($d = \{1, 2, 3, 4\}$) | Pakistan signed a resolution to import 1,300 MW of electricity. |

| Input | US whistleblower Bradley Manning, charged with releasing over 700,000 battlefield reports from Iraq and Afghanistan to Wikileaks, received a sentence of 35 years in prison from a military court Wednesday. |
|-------|-----------------------------------------------------------------------------------------------------------|
| Gold  | Bradley Manning received a sentence of 35 years in prison. |
| Tagger| Bradley Manning received a sentence of 35 years. |
| Tagger-ILP | Bradley Manning received a sentence of years. |
| Base  | Bradley Manning received a sentence of 35 years. |
| HiSAN-Dep ($d = \{1\}$) | Bradley Manning charged with releasing over 700,000 battlefield reports to Wikileaks received. |
| HiSAN w/ syn ($d = \{1, 2, 4\}$) | Bradley Manning received a sentence of 35 years in prison. |
| HiSAN w/o syn ($d = \{1, 2, 3, 4\}$) | Bradley Manning received a sentence of 35 years in prison. |

**Table 5** Example sentences and compressions.

**Tagger** match or exceed those of the Seq2Seq-based methods. The compressed sentence in the first example in Table 5 generated by **Tagger** is ungrammatical. We believe that this is mainly because **Tagger** cannot consider the predicted labels of the previous words. **Tagger-**
ILP outputs grammatically incorrect compressed sentences in both examples, which indicates that the ILP constraint imposed on the parent-child relationships between words is insufficient to generate fluent sentences. Compared with these baselines, HiSAN w/syn ($d = \{1, 2, 4\}$) output fluent and more informative compressed sentences. HiSAN w/o syn ($d = \{1, 2, 3, 4\}$) also output fluent compressed sentences. However, in the first example, the output of HiSAN w/o syn ($d = \{1, 2, 3, 4\}$) dropped the country names. This may indicate that HiSAN w/o syn ($d = \{1, 2, 3, 4\}$) did not capture the relationship between the children of “signed” owing to a lack of syntactic dependency information. These observations, which confirmed our expectation that syntactic dependency information helps in retaining important words, is supported by the automatic and human evaluation results.

We confirm that the compression performance of HiSAN w/syn actually improves if the sentences have deep dependency trees. Table 6 shows the automatic evaluation results for sentences with deep dependency trees. We can observe that HiSANs with higher-order dependency chains has better compression performance if the sentences have deep dependency trees. In terms of F1 scores, HiSAN w/syn $d = \{1, 2, 4\}$ outperformed HiSAN w/o syn $d = \{1, 2, 3, 4\}$, which indicates that explicit syntactic dependency information can help compress sentences with deep dependency trees.

Fig. 11 shows a compressed sentence and its dependency graph as determined by HiSAN w/syn $d = \{1, 2, 4\}$. Almost all the arcs with large probabilistic weights are contained in the parsed dependency trees. Interestingly, a few arcs not contained in the parsed dependency trees directly connect words that are connected by the dependency chains (bold gray arrows) in the original parsed dependency tree.

Considering that the training dataset does not contain such dependency relationships, we can estimate that these arcs are learned for supporting the sentence compression. This result meets our expectation that the dependency chain information is necessary for compressing sentences accurately.

We present a matrix style visualization of the attention distribution for HiSAN w/o syn. Different from HiSAN w/syn, HiSAN w/o syn displays smooth distributions owing to the lack of supervised dependency tree information. Therefore, visualizing them with the dependency style shown in Fig. 11 is unsuitable for their interpretation. Fig. 12 shows the attention distributions for each setting of HiSAN w/o syn. In the matrices in $d = \{1\}$, $d = \{1, 2\}$, $d = \{1, 2, 4\}$, and $d = \{1, 2, 3, 4\}$, we did not observe any obvious sentence head or tree-like relationships between the parent and child words. In the case of $d = \{1, 2, 3\}$, it is evident that the two to-clauses are treated as the heads of other words. However, the verb “signed” is less attended than other words.
Table 6  Results of automatic evaluation using sentences with deep dependency trees (deeper than average depth, 8). The notations are the same as in Table 2.

Table 6:

|        | F1       | ROUGE 1 | ROUGE 2 | ROUGE L | ΔC   | CR   |
|--------|----------|---------|---------|---------|------|------|
| Gold   |          |         |         |         |      | 34.2 |
| Tagger | 82.8 ± 0.4 | 76.0 ± 0.6 | 68.3 ± 0.8 | 76.1 ± 0.8 | −3.2 ± 0.9 | 31.0 ± 0.9 |
| Tagger+ILP | 77.5 ± 0.3 | 68.8 ± 0.5 | 60.2 ± 0.5 | 70.2 ± 0.4 | −4.6 ± 0.5 | 29.6 ± 0.5 |
| Bi-LSTM | 81.3 ± 0.7 | 75.1 ± 1.2 | 69.1 ± 1.3 | 75.7 ± 1.3 | −2.3 ± 0.7 | 31.9 ± 0.7 |
| Bi-LSTM-Dep | 81.5 ± 0.6 | 75.8 ± 0.6 | 69.8 ± 0.8 | 76.3 ± 0.8 | −2.1 ± 0.7 | 32.1 ± 0.7 |
| Attn   | 81.9 ± 0.6 | 76.0 ± 0.7 | 70.2 ± 1.1 | 76.7 ± 0.9 | −2.3 ± 0.4 | 31.9 ± 0.4 |
| Base   | 82.1 ± 0.7 | 76.4 ± 1.0 | 70.3 ± 1.3 | 76.7 ± 1.1 | −2.6 ± 0.6 | 31.6 ± 0.6 |
| HiSAN-Dep d = {1} | 82.3 ± 0.6 | 76.5 ± 0.9 | 70.7 ± 1.0 | 77.0 ± 0.9 | −2.5 ± 0.6 | 31.7 ± 0.6 |
| HiSAN-Dep d = {1, 2} | 81.9 ± 0.7 | 76.2 ± 0.6 | 70.4 ± 0.8 | 76.7 ± 0.7 | −2.7 ± 0.5 | 31.5 ± 0.5 |
| HiSAN-Dep d = {1, 2, 3} | 82.6 ± 0.8 | 76.5 ± 1.1 | 70.8 ± 1.2 | 77.0 ± 1.1 | −2.7 ± 0.5 | 31.5 ± 0.5 |
| HiSAN-Dep d = {1, 2, 3, 4} | 82.0 ± 0.4 | 76.2 ± 0.4 | 70.2 ± 0.4 | 76.6 ± 0.4 | −2.7 ± 0.4 | 31.5 ± 0.4 |
| HiSAN w/ syn d = {1} | 82.7 ± 0.5 | 76.6 ± 0.7 | 70.8 ± 0.5 | 77.1 ± 0.5 | −2.8 ± 0.4 | 31.4 ± 0.4 |
| HiSAN w/ syn d = {1, 2} | 82.6 ± 0.6 | 77.0 ± 0.5 | 71.3 ± 0.5 | 77.6 ± 0.5 | −2.2 ± 0.7 | 32.0 ± 0.7 |
| HiSAN w/ syn d = {1, 2, 3} | 82.6 ± 0.4 | 77.0 ± 0.4 | 71.4 ± 0.5 | 77.6 ± 0.5 | −2.3 ± 0.4 | 31.9 ± 0.4 |
| HiSAN w/ syn d = {1, 2, 3, 4} | 82.8 ± 0.4 | 77.0 ± 0.9 | 71.5 ± 1.1 | 77.7 ± 1.0 | −2.1 ± 0.7 | 32.1 ± 0.7 |
| HiSAN w/o syn d = {1} | 82.4 ± 1.0 | 76.6 ± 1.2 | 70.8 ± 1.7 | 77.2 ± 1.5 | −2.5 ± 0.5 | 31.7 ± 0.5 |
| HiSAN w/o syn d = {1, 2} | 82.3 ± 0.4 | 76.6 ± 0.3 | 70.6 ± 0.5 | 77.0 ± 0.3 | −2.8 ± 0.7 | 31.6 ± 0.7 |
| HiSAN w/o syn d = {1, 2, 3} | 82.2 ± 0.5 | 76.4 ± 0.2 | 70.6 ± 0.3 | 76.9 ± 0.2 | −2.5 ± 0.2 | 31.7 ± 0.2 |
| HiSAN w/o syn d = {1, 2, 3, 4} | 82.3 ± 0.5 | 76.4 ± 1.0 | 70.6 ± 1.2 | 76.9 ± 1.1 | −2.4 ± 0.8 | 31.8 ± 0.8 |
| HiSAN w/o syn d = {1, 2, 3, 4}* | 82.2 ± 0.8 | 76.3 ± 0.8 | 70.4 ± 1.1 | 76.8 ± 1.1 | −2.6 ± 0.9 | 31.6 ± 0.9 |

Fig. 11 An example compressed sentence and its dependency graph with HiSAN d = {1, 2, 4}. The gray-colored words represent deleted words. The numbers for each arc represent the probabilistic weight of a relationship between parent and child words. The arcs contained in the parsed dependency tree are located on the top side. The arcs not contained in the parsed dependency tree are located at the bottom.
Fig. 12  Attention distributions for each setting of HiSAN w/o syn. The words in columns represent an input sentence while those in rows represent the compressed sentences for each method. The gray-colored words represent deleted words.
even though the word is the head of the gold compressed sentence. These tendencies indicate that the attention distributions obtained through unsupervised training are unlikely to contain explicit syntactic dependency structures. This can be a reason why $F_1$ scores of HiSAN w/o syn were largely lower than those of HiSAN w/ syn in the sentences with deep dependency trees.

Table 7 shows the $F_1$ scores of HiSAN $d = \{1, 2, 4\}$ for each $\lambda$. The model with the setting $\lambda = 1.0$ or $\lambda = 0.0$ achieved better $F_1$ scores compared with those obtained with the other settings of $\lambda$. As we had discussed during the attention distribution analysis, $\lambda = 1.0$ and $\lambda = 0.0$ have largely different attention distributions. Because HiSAN contains only a single self-attention module, it is difficult to express such multiple attention distributions in the same network. Therefore, we still have room to improve the compression performance by using recursive attention in multi-head attention (Vaswani et al. 2017).

### 6 Related Work

Several neural network-based methods for sentence compression use syntactic features. Filippova et al. (2015) employed the features obtained from automatic parse trees in the LSTM-based encoder-decoder in a pipeline manner. Wang et al. (2017) trimmed the dependency trees based on the scores predicted by an LSTM-based tagger. Although these methods can consider the dependency relationships between words, the pipeline approach and the first-order dependency relationship fail to compress sentences that are longer than average.

Several machine translation studies have also utilized syntactic features in Seq2Seq models. Eriguchi et al. (2017) and Aharoni and Goldberg (2017) incorporated syntactic features of the target language in the decoder part of Seq2Seq. Both methods outperformed Seq2Seq without syntactic features in terms of translation quality. Song et al. (2020) adopted the model of Eriguchi et al. (2017) for headline generation tasks and reported improvements in summarization quality. However, these methods fail to provide an entire parse tree until the decoding phase is finished. Thus, these methods cannot track all the possible parents for each word within the decoding process. Similar to HiSAN, Hashimoto and Tsuruoka (2017) used dependency features

| $\lambda$ | Dev     | Test  |
|-----------|---------|-------|
| $\lambda = 1.0$ | 84.2 ± 0.2 | **83.2 ± 0.3** |
| $\lambda = 0.1$ | 84.1 ± 0.3 | 82.7 ± 0.3 |
| $\lambda = 0.01$ | 84.0 ± 0.4 | 82.3 ± 0.3 |
| $\lambda = 0.0$ | **84.3 ± 0.3** | 82.7 ± 0.3 |

Table 7 $F_1$ scores of HiSAN $d = \{1, 2, 4\}$ in the development and test datasets for each $\lambda$. 
as attention distributions; however, different from HiSAN, they used pre-trained dependency relations, and did not take into account the chains of dependencies. They reported an improvement in BLEU scores when their Seq2Seq was trained with syntactic dependency information. Marcheggiani and Titov (2017) and Bastings et al. (2017) considered higher-order dependency relationships in Seq2Seq by incorporating a graph convolution technique (Kipf and Welling 2016) into the encoder. However, the dependency information of the graph convolution technique was still provided in a pipeline manner, and thus, their method cannot work without syntactic tree information. This weak point restricts the availability of their model when a target dataset is not accompanied by syntactic trees. Such a case is possible with out-of-domain datasets and low resourced languages.

Unlike the above methods, HiSAN can capture higher-order dependency features using $d$-length dependency chains without relying on pipeline processing. In addition, it can continue learning even in the absence of syntactic dependency information. Recently, Zhao et al. (2018) also proposed a method that can avoid the effect of parse fails by incorporating a syntax-based language model into a sequential tagger as a reward for reinforcement learning. Different from HiSAN, their method requires a language model to run during training. Niu et al. (2019) utilized a pre-trained language model BERT (Devlin et al. 2019) to generate a grammatically compressed sentences in an unsupervised manner. For considering the successful use of such a language model, we extended HiSAN to incorporate ELMo (Peters et al. 2018) and BERT for capturing important words (Kamigaito and Okumura 2020). In addition, this extension included a new decoding method not considered in HiSAN. This decoding method can consider words that will be decoded in the future for compressing sentences more accurately. Furthermore, HiSAN was also extended to solve single document summarization (Ishigaki et al. 2019). The success of these extensions shows the versatility of HiSAN.

7 Conclusion

In this study, we investigated the performance of HiSAN, our proposed model that incorporates higher-order dependency features into Seq2Seq to compress sentences of all lengths.

Experiments on the Google sentence compression test data showed that HiSAN achieved better results than baseline methods on $F_1$ as well as ROUGE-1, -2, and -L scores (83.2, 78.6, 71.6, and 78.3, respectively). Particularly, when challenged with longer than average sentences, HiSAN outperformed the baseline methods in terms of $F_1$ and ROUGE-1, -2, and -L scores. HiSAN also outperformed the previous methods in both readability and informativeness during
human evaluations. Furthermore, HiSAN outperformed the baseline methods without using any syntactic dependency information.

Our investigation shows the effectiveness of syntactic dependency information in compressing long sentences with higher F1 scores by maintaining accurate compression rates. Furthermore, our analysis also indicates that HiSAN can compress sentences with precise length without relying on any syntactic dependency tree. From these results, we conclude that HiSAN is a useful tool for sentence compression tasks.

Acknowledgement

This paper is an extended version of our preliminary paper (Kamigaito et al. 2018) presented in NAACL-HLT 2018. In this paper, we add detailed explanations and the following new information to the original one: Investigation regarding the performance of HiSAN without syntactic information; the list of parameters for each compared method; standard deviations for each score of the automatic evaluation; statistics for the human evaluation dataset; discussion of the automatic evaluation results for each sentence length; discussion of the attention distributions for HiSANS; scores of HiSAN for each $\lambda$; related work including recent research.

References

Aharoni, R. and Goldberg, Y. (2017). “Towards String-To-Tree Neural Machine Translation.” In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pp. 132–140, Vancouver, Canada. Association for Computational Linguistics.

Bastings, J., Titov, I., Aziz, W., Marcheggiani, D., and Simaan, K. (2017). “Graph Convolutional Encoders for Syntax-aware Neural Machine Translation.” In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 1957–1967, Copenhagen, Denmark. Association for Computational Linguistics.

Berg-Kirkpatrick, T., Gillick, D., and Klein, D. (2011). “Jointly Learning to Extract and Compress.” In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pp. 481–490, Portland, Oregon, USA. Association for Computational Linguistics.

Chen, Y. and Pan, R. (2017). “Automatic Emphatic Information Extraction from Aligned Acoustic Data and Its Application on Sentence Compression.” In Proceedings of the 31st AAAI
Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. (2014). “On the Properties of Neural Machine Translation: Encoder–Decoder Approaches.” In Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, pp. 103–111, Doha, Qatar. Association for Computational Linguistics.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Eriguchi, A., Tsuruoka, Y., and Cho, K. (2017). “Learning to Parse and Translate Improves Neural Machine Translation.” In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pp. 72–78, Vancouver, Canada. Association for Computational Linguistics.

Filippova, K., Alfonseca, E., Colmenares, C. A., Kaiser, L., and Vinyals, O. (2015). “Sentence Compression by Deletion with LSTMs.” In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 360–368, Lisbon, Portugal. Association for Computational Linguistics.

Filippova, K. and Altun, Y. (2013). “Overcoming the Lack of Parallel Data in Sentence Compression.” In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 1481–1491, Seattle, Washington, USA. Association for Computational Linguistics.

Glorot, X. and Bengio, Y. (2010). “Understanding the Difficulty of Training Deep Feedforward Neural Networks.” In Proceedings of the 13th International Conference on Artificial Intelligence and Statistics, pp. 249–256.

Hashimoto, K. and Tsuruoka, Y. (2017). “Neural Machine Translation with Source-Side Latent Graph Parsing.” In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 125–135, Copenhagen, Denmark. Association for Computational Linguistics.

Ishigaki, T., Kamigaito, H., Takamura, H., and Okumura, M. (2019). “Discourse-Aware Hierarchical Attention Network for Extractive Single-Document Summarization.” In Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019), pp. 497–506, Varna, Bulgaria. INCOMA Ltd.
Jing, H. (2000). “Sentence Reduction for Automatic Text Summarization.” In *Proceedings of the 6th Conference on Applied Natural Language Processing*, pp. 310–315, Seattle, Washington, USA. Association for Computational Linguistics.

Kamigaito, H., Hayashi, K., Hirao, T., and Nagata, M. (2018). “Higher-Order Syntactic Attention Network for Longer Sentence Compression.” In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1716–1726, New Orleans, Louisiana. Association for Computational Linguistics.

Kamigaito, H., Hayashi, K., Hirao, T., Takamura, H., Okumura, M., and Nagata, M. (2017). “Supervised Attention for Sequence-to-Sequence Constituency Parsing.” In *Proceedings of the 8th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 7–12, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Kamigaito, H. and Okumura, M. (2020). “Syntactically Look-Ahead Attention Network for Sentence Compression.” *arXiv preprint arXiv:2002.01145*.

Kingma, D. P. and Ba, J. (2014). “Adam: A Method for Stochastic Optimization.” *arXiv preprint arXiv:1412.6980*.

Kipf, T. N. and Welling, M. (2016). “Semi-Supervised Classification with Graph Convolutional Networks.” *arXiv preprint arXiv:1609.02907*.

Klerke, S., Goldberg, Y., and Søgaard, A. (2016). “Improving Sentence Compression by Learning to Predict Gaze.” In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1528–1533, San Diego, California. Association for Computational Linguistics.

Knight, K. and Marcu, D. (2000). “Statistics-based Summarization-step One: Sentence Compression.” *AAAI/IAAI, 2000*, pp. 703–710.

Koehn, P. and Knowles, R. (2017). “Six Challenges for Neural Machine Translation.” In *Proceedings of the 1st Workshop on Neural Machine Translation*, pp. 28–39, Vancouver. Association for Computational Linguistics.

Lin, C.-Y. and Och, F. J. (2004). “Automatic Evaluation of Machine Translation Quality Using Longest Common Subsequence and Skip-Bigram Statistics.” In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL’04), Main Volume*, pp. 605–612, Barcelona, Spain.

Luong, T., Pham, H., and Manning, C. D. (2015). “Effective Approaches to Attention-based Neural Machine Translation.” In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1412–1421, Lisbon, Portugal. Association for Compu-
Marcheggiani, D. and Titov, I. (2017). “Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling.” In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 1506–1515, Copenhagen, Denmark. Association for Computational Linguistics.

Neubig, G., Dyer, C., Goldberg, Y., Matthews, A., Ammar, W., Anastasopoulos, A., Ballesteros, M., Chiang, D., Clothiaux, D., Cohn, T., et al. (2017). “DyNet: The Dynamic Neural Network Toolkit.” arXiv preprint arXiv:1701.03980.

Niu, T., Xiong, C., and Socher, R. (2019). “Deleter: Leveraging BERT to Perform Unsupervised Successive Text Compression.” arXiv preprint arXiv:1909.03223.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). “Deep Contextualized Word Representations.” In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pp. 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Pouget-Abadie, J., Bahdanau, D., van Merrienboer, B., Cho, K., and Bengio, Y. (2014). “Overcoming the Curse of Sentence Length for Neural Machine Translation using Automatic Segmentation.” arXiv preprint arXiv:1409.1257.

Song, K., Lebanoff, L., Guo, Q., Qiu, X., Xue, X., Li, C., Yu, D., and Liu, F. (2020). “Joint Parsing and Generation for Abstractive Summarization.” In AAAI 2020, pp. 8894–8901.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). “Dropout: A Simple Way to Prevent Neural Networks from Overfitting.” The Journal of Machine Learning Research, 15 (1), pp. 1929–1958.

Tran, N.-T., Luong, V.-T., Nguyen, N. L.-T., and Ngkiem, M.-Q. (2016). “Effective Attention-based Neural Architectures for Sentence Compression with Bidirectional Long Short-term Memory.” In Proceedings of the 7th Symposium on Information and Communication Technology, SoICT ’16, pp. 123–130, New York, NY, USA. ACM.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). “Attention is All you Need.” In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (Eds.), Advances in Neural Information Processing Systems, Vol. 30, pp. 5998–6008. Curran Associates, Inc.

Vinyals, O., Kaiser, L. u., Koo, T., Petrov, S., Sutskever, I., and Hinton, G. (2015). “Grammar as a Foreign Language.” In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (Eds.), Advances in Neural Information Processing Systems 28, pp. 2773–2781.
Wang, L., Jiang, J., Chieu, H. L., Ong, C. H., Song, D., and Liao, L. (2017). “Can Syntax Help? Improving an LSTM-based Sentence Compression Model for New Domains.” In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1385–1393, Vancouver, Canada. Association for Computational Linguistics.

Yao, K. and Zweig, G. (2015). “Sequence-to-sequence Neural Net Models for Grapheme-to-Phoneme Conversion.” In *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6–10, 2015*, pp. 3330–3334.

Zewei, S., Shujian, H., Hao-Ran, W., Xinyu, D., and Jiajun, C. (2020). “Generating Diverse Translation by Manipulating Multi-Head Attention.” In *The 34th AAAI Conference on Artificial Intelligence, AAAI 2020, The 32nd Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The 10th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7–12, 2020*, pp. 8976–8983. AAAI Press.

Zhang, X., Cheng, J., and Lapata, M. (2017). “Dependency Parsing as Head Selection.” In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pp. 665–676, Valencia, Spain. Association for Computational Linguistics.

Zhao, Y., Luo, Z., and Aizawa, A. (2018). “A Language Model based Evaluator for Sentence Compression.” In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 170–175, Melbourne, Australia. Association for Computational Linguistics.

**Hidetaka Kamigaito**: received his Ph.D. from Tokyo Institute of Technology, and is currently an assistant professor at Institute of Innovative Research, Tokyo Institute of Technology. His current research interests are in natural language processing with a specific focus on document-level text processing.

**Katsuhiko Hayashi**: received PhD in Engineering from the graduate school of Information Science at the Nara Institute of Science and Technology in 2013. He is currently an associate professor in Gunma University and a visiting researcher in the Riken Center for Advanced Intelligence Project. His research
interests include natural language processing and statistical relational learning.

**Tsutomu Hirao:** received the B.E. from Kansai University in 1995 and the M.E. and Ph.D. in engineering from Nara Institute of Science and Technology in 1997 and 2002, respectively. He is currently with NTT Communication Science Laboratories. His current research interests include Natural Language Processing and Machine Learning.

**Masaaki Nagata:** received the B.E., M.E., and Ph.D. degrees in information science from Kyoto University, Japan, in 1985, 1987, and 1999, respectively. In 1987, he joined Nippon Telegraph and Telephone Corporation (NTT). He is currently a Senior Distinguished Researcher at NTT Communication Science Laboratories. He is a member of the Association for Computational Linguistics and the Association for Natural Language Processing.

**Manabu Okumura:** received Dr. Eng. from Tokyo Institute of Technology, and is currently a professor at Institute of Innovative Research, Tokyo Institute of Technology. His current research interests include natural language processing, especially text summarization, computer assisted language learning, sentiment analysis, and text data mining.

(Received August 16, 2020)
(Revised December 1, 2020)
(Accepted January 14, 2021)