Syntactically Look-Ahead Attention Network for Sentence Compression

Hidetaka Kamigaito, Manabu Okumura
Institute of Innovative Research, Tokyo Institute of Technology
kamigaito@lr.pi.titech.ac.jp, oku@pi.titech.ac.jp

Abstract
Sentence compression is the task of compressing a long sentence into a short one by deleting redundant words. In sequence-to-sequence (Seq2Seq) based models, the decoder unidirectionally decides to retain or delete words. Thus, it cannot usually explicitly capture the relationships between decoded words and unseen words that will be decoded in the future time steps. Therefore, to avoid generating ungrammatical sentences, the decoder sometimes drops important words in compressing sentences. To solve this problem, we propose a novel Seq2Seq model, syntactically look-ahead attention network (SLAHAN), that can generate informative summaries by explicitly tracking both dependency parent and child words during decoding and capturing important words that will be decoded in the future. The results of the automatic evaluation on the Google sentence compression dataset showed that SLAHAN achieved the best kept-token-based-F1, ROUGE-1, ROUGE-2 and ROUGE-L scores of 85.5, 79.3, 71.3 and 79.1, respectively. SLAHAN also improved the summarization performance on longer sentences. Furthermore, in the human evaluation, SLAHAN improved informativeness without losing readability.

Introduction
Sentence compression is the task of producing a shorter sentence by deleting words in the input sentence while preserving its grammaticality and important information. To compress a sentence so that it is still grammatical, tree trimming methods (Jing 2000; Knight and Marcu 2000; Berg-Kirkpatrick, Gillick, and Klein 2011; Filippova and Altun 2013) have been utilized. However, these methods often suffer from parsing errors. As an alternative, Filippova et al. (2015) proposed a method based on sequence-to-sequence (Seq2Seq) models that do not rely on parse trees but produce fluent compression. However, the vanilla Seq2Seq model has a problem that it is not so good for compressing longer sentences.

To solve the problem, Kamigaito et al. (2018) expanded Seq2Seq models to capture the relationships between long-distance words through recursively tracking dependency parents from a word with their recursive attention module. Their model learns dependency trees and compresses sentences jointly to avoid the effect of parsing errors. This improvement enables their model to compress a sentence while preserving the important words and its fluency.

However, since their method focuses only on parent words, important child words of the currently decoded word would be lost in compressed sentences. That is, in Seq2Seq models, because the decoder unidirectionally compresses sentences, it cannot usually explicitly capture the relationships between decoded words and unseen words which will be decoded in the future time steps. As the result, to avoid producing ungrammatical sentences, the decoder sometimes drops important words in compressing sentences. To solve the problem, we need to track both parent and child words to capture unseen important words that will be decoded in the future time steps.

Fig.1 shows an example of sentence compression that needs to track both parent and child words. Since the input sentence mentions the export of the plane between two countries, we have to retain the name of the plane, import country and export country in the compressed sentence. When the decoder reads “Japan”, it should recursively track both the parent and child words of “Japan”. Then, it can decide to retain “hold” that is the parent of “Japan” and the syntactic head of the sentence. By retaining “hold” in the compressed sentence, it can also retain “Japan”, “and” and “India” because these are the child and grandchild of “hold” (the top case in Fig.1).

When the decoder reads “hold”, it should find the important phrase “Japan’s export” by recursively tracking child words from “hold”. The tracking also supports the decoder for retaining “talks” and “on” to produce grammatical compression (the middle case).

When the decoder reads “export”, it should track child words to find the important phrase “US2 rescue plane” and retain “of” for producing grammatical compression (the bottom case).

Note that a decoder that tracks only parent words cannot find the important phrases or produce grammatical compress-

1This sentence actually belongs to the test set of the Google sentence compression dataset (https://github.com/google-research-datasets/sentence-compression).
Japan and India will hold working-level talks here Wednesday on Japan’s export of US2 rescue plane to India.

Figure 1: An example sentence and its dependency tree during the decoding process. The gray words represent deleted words, and the words in black frames are currently decoded words. Already decoded words are underlined. The tracking of parent nodes is represented as blue edges, and that of child nodes is represented as red edges. The bold words represent the important words in this sentence.

Figure 2: The proportion of words retained later that are linked from right to the retained words in the summary as a parent or a child word in the left-to-right decoding.

Our Base Seq2Seq Model
Sentence compression is a kind of text generation task. However, it can also be considered as a sequential tagging task, where given a sequence of input tokens \( x = (x_0, ..., x_n) \), a sentence summarizer predicts an output label \( y_t \) from specific labels (“keep”, “delete” or “end of a sentence”) for each corresponding input token \( x_t \) \((1 ≤ t ≤ n)\). Note that \( x_0 \) is the start symbol of a sentence.

To generate a grammatically correct summary, we choose Seq2Seq models as our base model. For constructing a robust baseline model, we introduce recently proposed contextualized word embeddings such as ELMo (Peters et al. 2018) and BERT (Devlin et al. 2018) into the sentence compression task. As described later in our evaluation results, this baseline exceeds the state-of-the-art \( F_1 \) scores reported by Zhao, Luo, and Aizawa (2018).

Our base model consists of embedding, encoder, decoder, and output layers. In the embedding layer, the model extracts features from an input token \( x_i \) as a vector \( e_i \) as follows:

\[
e_i = \| F \|_{j=1} F_{i,j},
\]

where \( \| \) represents the vector concatenation, \( F_{i,j} \) is a vector of the \( j \)-th feature for token \( x_i \), and \( |F| \) is the number of features (at most 3). We choose features from GloVe (Pennington, Socher, and Manning 2014), ELMo or BERT vectors. Because ELMo and BERT have many layers, we treat their weighted sum as \( F_{i,j} \) as follows:

\[
F_{i,j} = \sum_{k=1}^{(|L|)} \psi_{j,k} \cdot L_{i,j,k},
\]

\[
\psi_{j,k} = \exp(\phi_{j,k} \cdot L_{i,j,k}) / \sum_{l=1}^{(|L|)} \exp(\phi_{l,j} \cdot L_{i,l,k}),
\]

where \( L_{i,j,k} \) represents the \( k \)-th layer of the \( j \)-th feature for the token \( x_i \), and \( \phi_{j,k} \) is the weight vector for the \( k \)-th layer of the \( j \)-th feature. In BERT, to align the input token and the output label, we treat the average of sub-word vectors as a single word vector.

The encoder layer first converts \( e_i \) into a hidden state \( \vec{h}_i = LSTM(\vec{h}_{i-1}, e_i) \) by using forward-LSTM, and \( \vec{h}_i \) is calculated similarly by using backward-LSTM. Secondly, \( \vec{h}_i \) and \( \hat{h}_i \) are concatenated as \( h_i = [\vec{h}_i, \hat{h}_i] \). Through this process, the encoder layer converts the embedding \( e \) into a sequence of hidden states:

\[
h = (h_0, ..., h_n).
\]
The final state of the backward LSTM $\vec{h}_0$ is inherited by the decoder as its initial state.

At time step $t$, the decoder layer encodes the concatenation of a 3-bit one-hot vector determined by the predicted label $y_{t-1}$, the final hidden state $d_{t-1}$ (which we will explain later), and the token embedding $e_t$ into the decoder hidden state $\vec{s}_t$, by using a forward-LSTM.

The output layer predicts an output label probability as follows:

$$P(y_t \mid y_{<t}, x) = \text{softmax}(W_d d_t) \cdot \delta_{y_t},$$  \hspace{1cm}$d_t = \tanh(W_d[h_t, \vec{s}_t] + b_d), \quad (4)$

where $W_d$ is the weight matrix, $b_d$ is the bias term, $W_o$ is the weight matrix of the softmax layer, and $\delta_{y_t}$ is the binary vector where the $y_t$-th element is set to 1 and the other elements are set to 0.

**Syntactically Look-Ahead Attention Network**

In this section, we first explain the graph representation for a dependency tree that is used in SLAHAN and then introduce its entire network structure and the modules inside it. Both the graph representation and network parameters are jointly updated, as described in the later section.

**Graph Representation of Dependency Relationships**

We explain the details of our representation for tracking parent and child words from a word in a dependency tree. As described in Hashimoto and Tsuruoka (2017), a dependency relationship can be represented as a weighted graph. Given sentence $x = (x_0, \ldots, x_n)$, the parent of each word $x_j$ is selected from $x$. We treat $x_0$ as a root node. We represent the probability of $x_j$ being the parent of $x_t$ in $x$ as $P_{\text{head}}(x_j \mid x_t, x)$. By using $P_{\text{head}}(x_j \mid x_t, x)$, Kamigaito et al. (2018) show that $\alpha_{d,t,j}$, a probability of $x_j$ being the $d$-th order parent of $x_t$, is calculated as follows:

$$\alpha_{d,t,j} = \frac{\sum_{k=0}^{D} \alpha_{1,k} \cdot \alpha_{d-1,k} \cdot \beta_{d,t} \cdot \beta_{1,k} \cdot \beta_{1,k,j} \cdot \beta_{d,t,j}}{P_{\text{head}}(x_j \mid x_t, x)} \quad (d=1, \ldots, D). \quad (5)$$

Because the 1st line of Eq.(5) is a definition of matrix multiplication, by using a matrix $A^d$, which satisfies $A^0_{i,j} = \alpha_{0,i,j}$, Eq.(5) is reformulated as follows:

$$A^d = A^{d-1} A^1. \quad (6)$$

We call $A^d$ the $d$-th parent graph hereafter.

We expand Eq.(6) to capture $d$-th child words of a word $x_j$. At first, we define $P_{\text{child}}(x_t \mid x_j, x)$, the probability of $x_t$ being a child of $x_j$ in $x$, $P_x(x_j = p)$, the probability of $x_j$ being a parent word in $x$, $P_x(x_t = c)$, the probability of $x_t$ being a child word in $x$, and $P_x(x_j, x_t)$, the probability of $x_j$ and $x_t$ having a link in $x$. Assuming the probability of words having a link is independent of each other, the following equations are satisfied:

$$P_x(x_t, x_j) = P_{\text{child}}(x_t \mid x_j, x) \cdot P_x(x_j = p), \quad P_x(x_j, x_t) = P_{\text{head}}(x_j \mid x_t, x) \cdot P_x(x_t = c). \quad (7)$$

This can be reformulated as follows:

$$P_{\text{child}}(x_t \mid x_j, x) = P_{\text{head}}(x_j \mid x_t, x) \cdot P_x(x_t = c) / P_x(x_j = p). \quad (8)$$

Here, $P_x(x_t = c)$ is always 1 because of the dependency tree definition, and in this formulation, we treat $x_j$ as a parent; thus, $P_x(x_j = p)$ is a constant value. Therefore, we can obtain the following relationship:

$$P_{\text{child}}(x_t \mid x_j, x) \propto P_{\text{head}}(x_j \mid x_t, x). \quad (9)$$

Based on Eq.(9), we can define $\beta_{d,t,j}$, the strength of $x_j$ being the $d$-th order child of $x_t$, as follows:

$$\beta_{d,t,j} = \left\{ \begin{array}{ll} \sum_{k=0}^{D} \alpha_{1,k} \cdot \beta_{1,k} \cdot \beta_{d-1,k} \cdot \beta_{d,t} \cdot \beta_{1,k,j} \cdot \beta_{d,t,j} / P_{\text{head}}(x_j \mid x_t, x) \quad & (d=1) \\ 1 \quad & (d>1) \end{array} \right. \quad (10)$$

Similar to Eq.(5), by using a matrix $B^d$, which satisfies $B^0_{i,j} = \beta_{1,i,j}$, Eq.(10) is reformulated as follows:

$$B^d = B^{d-1} B^1. \quad (11)$$

We call $B^d$ the $d$-th child graph hereafter. Note that from the definition of the 2nd lines in Eq.(5) and Eq.(10), $A^1$ and $B^1$ always satisfy $B^1_{i,j} = A^1_{i,j}$. This can be reformulated as $B^1 = (A^1)^T$. Furthermore, from the definition of the transpose of a matrix, we can obtain the following formulation:

$$B^d = B^{d-1} B^1 = (A^1)^T (A^1)^T \cdots (A^1)^T = (A^d)^T. \quad (12)$$

Thus, once we calculate Eq.(6), we do not need to compute Eq.(11) explicitly. Therefore, letting $d$ be a dimension size of hidden vectors, the computational cost of SLAHAN is $O(n^2 d^2)$, similar to Kamigaito et al. (2018). This is based on the assumption that $d$ is larger than $n$ in many cases. Note that the computational cost of the base model is $O(n^3 d^3)$.

**Network Structure**

Fig.3 shows the entire structure of SLAHAN. It is constructed on our base model, as described in the previous section. After encoding the input sentence, the hidden states are passed to our network modules. The functions of each module are as follows:

**Head Attention** module makes a dependency graph of a sentence by calculating the probability of $x_j$ being the parent of $x_t$ based on $h_j$ and $h_t$ in Eq.(3) for each $x_t$.

**Parent Recurrent Attention** module calculates $d$-th parent graph $A^d$ and extracts a weighted sum of important hidden states $\mu_t^{\text{parent}}$ from $h$ in Eq.(3) based on $\alpha_{d,t,j}$ ($= A^d_{i,j}$) for each decoder time step $t$.

**Child Recurrent Attention** module uses $d$-th child graph $B^d$ to extract $\mu_t^{\text{child}}$, a weighted sum of important hidden states from $h$ in Eq.(3) based on $\beta_{d,t,j}$ ($= B^d_{i,j}$) for each decoder time step $t$.

**Selective Gate** module supports the decoder to capture important words that will be decoded in the future by calculating $\Omega_t$, the weighted sum of $\mu_t^{\text{parent}}$ and $\mu_t^{\text{child}}$, based on the current context. $\Omega_t$ is inherited to the decoder for deciding the output label $y_t$.

The details of each module are described in the following subsections.

**Head Attention** Similar to Zhang, Cheng, and Lapata (2017), we calculate $P_{\text{head}}(x_j \mid x_t, x)$ as follows:

$$P_{\text{head}}(x_j \mid x_t, x) = \text{softmax}(g(h_j, h_t)) \cdot \delta_{x_j}$$

$$g(h_j, h_t) = v^T \cdot \tanh(U \cdot h_j + W \cdot h_t), \quad (13)$$
where $v_a$, $U_a$ and $W_a$ are weight matrices of $g$. In a dependency tree, the root has no parent, and a token does not depend on itself. In order to satisfy these rules, we impose the following constraints on $P_{\text{head}}(x_j | x_t, x)$:

$$
P_{\text{head}}(x_j | x_t, x) = \begin{cases} 
1 & (t = 0 \land j = 0) \\
0 & (t = 0 \land j > 0) \\
0 & (t \neq 0 \land t = j).
\end{cases} \tag{14}
$$

The 1st and 2nd lines of Eq.(14) represent the case where the parent of root is also root. These imply that root does not have a parent. The 3rd line of Eq.(14) prevents a token from depending on itself. In the training phase, $P_{\text{head}}(x_j | x_t, x)$ is jointly learned with output label probability $P(y | x)$, as described in the objective function section.

**Parent Recursive Attention** The parent recursive attention module recursively calculates $\alpha_{d,t,j}$ by using $P_{\text{head}}(x_j | x_t, x)$ based on Eq.(5). The calculated $\alpha_{d,t,j}$ is used to weight the bi-LSTM hidden layer $h$ as follows:

$$
\gamma_{d,t} = \sum_{j=1}^{n} \alpha_{d,t,j} \cdot h_j. \tag{15}
$$

To select suitable dependency order $d$ for the input sentence, $\gamma_{d,t}$ is further weighted and summed to $\mu_{d,t}^{\text{parent}}$ by using weighting parameter $\eta_{d,t}$, according to the current context as follows:

$$
\begin{align*}
\epsilon_t &= \{ h_0, h_n, h_t, \gamma_t \}, \\
\eta_{d,t} &= \operatorname{softmax}(\gamma_{d,t} W_p^{\text{parent}} \cdot \epsilon_t) \cdot \delta_t, \\
\mu_{d,t}^{\text{parent}} &= \sum_{d=0}^{d-1} d \cdot \eta_{d,t},
\end{align*} \tag{16}
$$

where $W_p^{\text{parent}}$ is the weight matrix, $d$ is the group of dependency orders, and $\epsilon_t$ is the vector representing the current context.

**Child Recursive Attention** The child recursive attention module weights the bi-LSTM hidden layer $h$ based on $d$-th child graph $B^d$. Unlike the parent recursive attention module, $B^d$ is not a probability, and a word sometimes has more than two children. For that reason, we use max-pooling rather than the attention distribution in Eq.(15). In the child recursive attention module, the bi-LSTM hidden layer $h$ is weighted by $\beta_{d,t,j}$ and then pooled as follows:

$$
\rho_{d,t} = \operatorname{MaxPool}(\mu_{d,t}^{\text{parent}} \cdot (\beta_{d,t,j} \cdot h_j))^T. \tag{17}
$$

To select suitable dependency order $d$ for the input sentence, $\rho_{d,t}$ is further weighted and summed to $\mu_{d,t}^{\text{child}}$ by using weighting parameter $\eta_{d,t}$, according to the current context as follows:

$$
\eta_{d,t} = \operatorname{softmax}(\rho_{d,t} W_c^{\text{child}} \cdot \delta_t), \\
\mu_{d,t}^{\text{child}} = \sum_{d=0}^{d-1} d \cdot \eta_{d,t},
$$

where $W_c^{\text{child}}$ is the weight matrix.

**Selective Gate** This module calculates $\Omega_t$, a weighted sum of parent information $\mu_{t}^{\text{parent}}$ and child information $\mu_{t}^{\text{child}}$. The weight is decided by a gate $z_t$ by considering whether $\mu_{t}^{\text{parent}}$ or $\mu_{t}^{\text{child}}$ is more important in the current context. Specifically, $\Omega_t$ is calculated as follows:

$$
\Omega_t = z_t \cdot \mu_{t}^{\text{parent}} + (1 - z_t) \cdot \mu_{t}^{\text{child}}, \\
z_t = \sigma(W_z \cdot [\mu_{t}^{\text{parent}}; \mu_{t}^{\text{child}}; \epsilon_t]), \tag{19}
$$

where $\sigma$ is the element-wise product, $\sigma$ is the sigmoid function, and $W_z$ is the weight matrix. Then, $d_t$ in Eq.(4) is replaced by a concatenated vector $d'_t = [h_t, \Omega_t, \gamma_t]$; furthermore, instead of $d_t$, $d'_t$ is also fed to the decoder input at $t + 1$.

**Objective Function** To alleviate the effect of parse errors, we jointly update dependency parent probability $P_{\text{head}}(x_j | x_t)$ and label probability $P(y | x)$ (Kamigaito et al. 2017). We denote the existence of an edge between parent word $w_j$ and child word $w_t$ on a dependency tree as $a_{t,j} = 1$. In contrast, we denote the absence of an edge as $a_{t,j} = 0$. By using these notations, our objective function is defined as follows:

$$
-\log P(y | x) = \lambda \cdot \sum_{j=1}^{m} \sum_{i=1}^{n} a_{t,j} \cdot \log a_{t,j}, \tag{20}
$$

where $\lambda$ is a hyper-parameter balancing the importance of output labels and parse trees in training steps. To investigate the importance of the syntactic information, we used $\lambda = 1.0$ for the with syntax (w/syn) setting and $\lambda = 0$ for the without syntax (w/o syn) setting.
Experiments

For comparing our proposed models with the baselines, we conducted both automatic and human evaluations. The following subsections describe the evaluation details.

Settings

Datasets We used the Google sentence compression dataset (Google dataset) (Filippova and Altun 2013) for our evaluations. To evaluate the performances on an out-of-domain dataset, we also used the Broadcast News Compression Corpus (BNC Corpus)\(^3\). The setting for these datasets is as follows:

Google dataset: Similar to the previous researches (Filippova et al. 2015; Tran et al. 2016; Wang et al. 2017; Kamigaito et al. 2018; Zhao, Luo, and Aizawa 2018), we used the first 1,000 sentences of comp-data.eval.json as the test set. We used the last 1,000 sentences of comp-data.eval.json as our development set. Following recent researches (Kamigaito et al. 2018; Zhao, Luo, and Aizawa 2018), we used all 200,000 sentences in sent-comp.train*.json as our training set. We also used the dependency trees contained in this dataset.

To investigate the summarization performances on long sentences, we additionally performed evaluations on 417 sentences that are longer than the average sentence length (= 27.04) in the test set.

BNC Corpus: This dataset contains spoken sentences and their summaries created by three annotators. To evaluate the compression performances on long sentences in the out-of-domain setting, we treated sentences longer than the average sentence length, 19.83, as the test set (595 sentences), and training was conducted with the Google dataset. Because this dataset does not contain any dependency parsing results, we parsed all sentences in this dataset by using the Stanford dependency parser\(^4\). In all evaluations, we report the average scores for three annotators.

Compared Models The baseline models are as follows. We used ELMo, BERT and GloVe vectors for all models in our experiments.

Tagger: This is a bi-LSTM tagger which is used in various sentence summarization researches (Klerke, Goldberg, and Søgaard 2016; Wang et al. 2017).

LSTM: This is an LSTM-based sentence summarizer, which was proposed by Filippova et al. (2015).

LSTM-Dep: This is an LSTM-based sentence summarizer with dependency features, called LSTM-Par-Pres in Filippova et al. (2015).

Base: Our base model explained in the 2nd section.

Attn: This is an improved attention-based Seq2Seq model with ConCat attention, described in Luong, Pham, and Manning (2015). To capture the context of long sentences, we also feed input embedding into the decoder, similar to the study of Filippova et al. (2015).

Parent: This is a variant of SLAHAN that does not have the child recursive attention module. This model captures only parent words, similar to HiSAN in the study of Kamigaito et al. (2018). For the fair comparisons, we left the gate layer in Eq.(19).

Our proposed models are as follows:

SLAHAN: This is our proposed model which is described in the 3rd section.

Child: This is a variant of SLAHAN that does not have the parent recursive attention module. Similar to Parent, we left the gate layer in Eq.(19).

Model Parameters We used GloVe (glove.840B.300d), 3-layers of ELMo and 12-layers of BERT (cased_L-12_H-768_A-12) as our features. We first investigated the best combination of GloVe, ELMo, and BERT vectors as shown in Table 1. Following this result, we used the combination of all of GloVe, ELMo and BERT for all models.

The dimensions of the LSTM layer and the attention layer were set to 200. The depth of the LSTM layer was set to 2. These sizes were based on the setting of the LSTM NER tagger with ELMo in the study of Peters et al. (2018). All parameters were initialized with Glorot and Bengio (2010)'s method. For all methods, we applied Dropout (Srivastava et al. 2014) to the input of the LSTM layers. All dropout rates were set to 0.3. We used Adam (Kingma and Ba 2014) with an initial learning rate of 0.001 as our optimizer. All gradients were averaged by the number of sentences in each mini-batch. The clipping threshold value for the gradients was set to 5.0. The maximum training epoch was set to 20. We used \{1, 2, 3, 4\} as \(d\) in Eq.(16) and Eq.(18). The maximum mini-batch size was set to 16, and the order of mini-batches was shuffled at the end of each training epoch. We adopted early stopping to the models based on maximizing per-sentence accuracy (i.e., how many summaries are fully reproduced) of the development data set.

To obtain a compressed sentence, we used greedy decoding, following the previous research (Kamigaito et al. 2018). We used Dynet (Neubig et al. 2017) to implement our neural networks\(^5\).

Automatic Evaluation

Evaluation Metrics In the evaluation, we used kept-token-based-\(F_1\) measures (\(F_1\)) for comparing to the previously reported scores. In this metric, precision is defined as the ratio of kept tokens that overlap with the gold summary, and recall is defined as the ratio of tokens in the gold summary that overlap with the system output summary. For more concrete evaluations, we additionally used ROUGE-1 (\(R-1\)), ROUGE-2 (\(R-2\)), and ROUGE-L (\(R-L\)) (Lin and Och)

\(\begin{array}{c|ccccccccc}
\text{Model} & \text{Glove} & \checkmark & \checkmark & \checkmark & \checkmark & \checkmark & \checkmark & \checkmark & \checkmark \\
\text{ELMo} & \checkmark & \checkmark & \checkmark & \checkmark & \checkmark & \checkmark & \checkmark & \checkmark & \checkmark \\
\text{BERT} & \checkmark & \checkmark & \checkmark & \checkmark & \checkmark & \checkmark & \checkmark & \checkmark & \checkmark \\
\hline
\text{F}_1 & 86.2 & 86.0 & 85.9 & 85.4 & 85.3 & 85.9 & 84.8 & \\
\end{array}
\)

Table 1: \(F_1\) scores for Base with various features in the development data. The bold score represents the highest score.

---

\(^3\)https://www.jamesclarke.net/research/resources

\(^4\)https://nlp.stanford.edu/software/sslahan

\(^5\)Our code will be available at https://github.com/kamigaito/sslahan
difference of the score from the best baseline (mostly Base) is statistically significant.

observe the score of SLAHAN w/ syn.

Parent w/o syn 85.3 78.3 70.4 78.1 -3.4 83.3 75.6 67.3 75.2 -3.4

Child w/o syn 85.4 78.8 70.7 78.5 -2.9 83.0 75.8 67.3 75.4 -3.0

Child w/o syn 85.2 78.6 70.8 78.4 -3.1 83.2 76.3 68.2 75.8 -2.8

SLAHAN w/o syn 85.4 78.0\(^{\dagger}\) 71.0\(^{\dagger}\) 78.6\(^{\dagger}\) -3.0 83.6 76.5\(^{\dagger}\) 68.5\(^{\dagger}\) 76.1\(^{\dagger}\) -2.9

Table 2: Results on the Google dataset. ALL and LONG represent, respectively, the results for all sentences and only for long sentences (longer than average length 27.04) in the test dataset. The bold values indicate the best scores. \(^{\dagger}\) indicates that the difference of the score from the best baseline (mostly Base) is statistically significant.\(^{8}\)

| Model | F\(_1\) | R-1 | R-2 | R-L | \(\Delta C\) |
|-------|-------|-----|-----|-----|---------|
| Tagger | 54.6 | 36.8 | 27.7 | 36.4 | -39.1 |
| LSTM | 54.8 | 36.6 | 28.0 | 36.2 | -39.2 |
| LSTM-Deg | 55.1 | 36.9 | 28.2 | 36.5 | -38.8 |
| Attn | 54.1 | 36.1 | 27.4 | 35.6 | -39.6 |
| Base | 54.4 | 37.4 | 28.5 | 36.9 | -38.6 |
| Parent w/ syn | 54.2 | 36.3 | 27.7 | 35.9 | -39.1 |
| Parent w/o syn | 54.0 | 35.8 | 27.2 | 35.4 | -40.1 |
| Child w/ syn | 55.6 | 37.8 | 28.5 | 37.3 | -38.2 |
| Child w/o syn | 54.8 | 36.7 | 28.1 | 36.3 | -39.2 |
| SLAHAN w/ syn | 57.1\(^{\dagger}\) | 40.1\(^{\dagger}\) | 30.6\(^{\dagger}\) | 39.6\(^{\dagger}\) | -35.9\(^{\dagger}\) |
| SLAHAN w/o syn | 54.6 | 36.4 | 27.8 | 36.0 | -39.5 |

Table 3: Results on the BNC Corpus. \(^{\dagger}\) indicates the same as in Table 2.

2004)\(^{6}\) with limitation by reference byte lengths\(^{7}\) as evaluation metrics. We used \(\Delta C = \text{system compression ratio} - \text{gold compression ratio}\) (Kamigaito et al. 2018) to evaluate how close the compression ratio of system outputs was to that of gold compressed sentences. Note that the gold compression ratios of all the sentences and the long sentences in the Google test set are respectively 43.7 and 32.4. Those of all the sentences and the long sentences in the BNC corpus are respectively 76.3 and 70.8. We used the macro-average for all reported scores. All scores are reported as the average scores of three randomly initialized trials.

Results Table 2 shows the evaluation results on the Google dataset. SLAHAN achieved the best scores on both all the sentences and the long sentences. Through these gains, we can understand that SLAHAN successfully captures important words by tracking both parent and child words. Child achieved better scores than Parent. This result coincides with our investigation that tracking child words is important especially for long sentences, as shown in Fig. 2. We can also observe the score of SLAHAN w/o syn is comparable to that of SLAHAN w/ syn. This result indicates that dependency graphs can work on the in-domain dataset without relying on given dependency parse trees.

We also show the evaluation results on the BNC corpus, the out-of-domain dataset, in Table 3. We can clearly observe that SLAHAN w/ syn outperforms other models for all metrics. Comparing between Base, Parent, Child and SLAHAN, we can understand that SLAHAN w/ syn captured important words during the decoding step even in the BNC corpus. The remarkable performance of SLAHAN w/ syn supports the effectiveness of explicit syntactic information. That is, in the out-of-domain dataset, the dependency graph learned with implicit syntactic information obtained lower scores than that learned with explicit syntactic information. The result agrees with the findings of the previous research (Wang et al. 2017). From these results, we can conclude that SLAHAN is effective even for both long and out-of-domain sentences.

Human evaluation

In the human evaluation, we compared the models\(^{8}\) that achieved the top five R-L scores in the automatic evaluation. We filtered out sentences whose compressions are the same for all the models and selected the first 100 sentences from the test set of the Google dataset. Those sentences were evaluated for both readability (Read) and informativeness (Info) by twelve raters, who were asked to rate them in a five-point Likert scale, ranging from one to five for each metric. To

\(^{6}\)We used the ROUGE-1.5.5 script with option “-n 2 -m –d -a”.

\(^{7}\)If a system output exceeds the reference summary byte length, we truncated the exceeding tokens.

\(^{8}\)We used paired-bootstrap-resampling (Koehn 2004) with 1,000,000 random samples \((p < 0.05)\).
In this paper, we proposed a novel Seq2Seq model, *syntactically look-ahead attention network* (SLAHAN), that can generate informative summaries by explicitly tracking parent and child words for capturing the important words in a sentence. The evaluation results showed that SLAHAN achieved the best kept-token-based-F1, ROUGE-1, ROUGE-2 and ROUGE-L scores on the Google dataset in both the sentence and the long sentence settings. In the...
BNC corpus, SLAHAN also achieved the best kept-token-based-F1, ROUGE-1, ROUGE-2 and ROUGE-L scores, and showed its effectiveness on both long sentences and out-of-domain sentences. In human evaluation, SLAHAN improved informativeness without losing readability. From these results, we can conclude that in Seq2Seq models, capturing important words that will be decoded in the future based on dependency relationships can help to compress long sentences during the decoding steps.

Acknowledgement
We are thankful to Dr. Tsutomu Hirao for his useful comments.

References
Akbik, A.; Bergmann, T.; and Vollgraf, R. 2019. Pooled contextualized embeddings for named entity recognition. 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), 724–728.

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, 481–490.

Clarke, J., and Lapata, M. 2007. Modelling compression with discourse constraints. In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), 1–11.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

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, 1481–1491.

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, 360–368.

Glorot, X., and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 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, 125–135.

Jing, H. 2000. Sentence reduction for automatic text summarization. In Proceedings of the Sixth Conference on Applied Natural Language Processing, 310–315.

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 Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers), 7–12.

Kamigaito, H.; Hayashi, K.; Hirao, T.; and Nagata, M. 2018. Higher-order syntactic attention network for long 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), 1716–1726.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. CoRR abs/1412.6980.

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, 1528–1533.

Knight, K., and Marcu, D. 2000. Statistics-based summarization step one: Sentence compression. AAAI/IAAI 703–710.

Koehn, P. 2004. Statistical significance tests for machine translation evaluation. In Proceedings of EMNLP 2004, 388–395.

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, 605–612.

Lin, X.; Joty, S.; Jwalapuram, P.; and Bari, M. S. 2019. A unified linear-time framework for sentence-level discourse parsing. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 4190–4200.

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, 1412–1421.

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.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In Empirical Methods in Natural Language Processing (EMNLP), 1532–1543.

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), 2227–2237.

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):1929–1958.

Tran, N.-T.; Luong, V.-T.; Nguyen, N. L.-T.; and Ng Hiem, M.-Q. 2016. Effective attention-based neural architectures for sentence compression with bidirectional long short-term memory. In Proceedings of the Seventh Symposium on Information and Communication Technology, 123–130.

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), 1385–1393.

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, 665–676.

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), 170–175.
# Appendix

Results on all sentences of BNC Corpus

|                | F₁   | R-1  | R-2  | R-L  | ΔC   |
|----------------|------|------|------|------|------|
| Tagger         | 68.4 | 56.7 | 44.4 | 56.5 | -24.6|
| LSTM           | 67.4 | 54.8 | 42.7 | 54.6 | -27.0|
| LSTM-Dep       | 68.0 | 55.6 | 43.7 | 55.3 | -26.2|
| Attn           | 67.1 | 54.3 | 43.1 | 54.1 | -26.5|
| Base           | 68.3 | 56.0 | 43.9 | 55.8 | -25.6|
| Parent w/ syn  | 67.7 | 55.7 | 43.7 | 55.5 | -25.8|
| Parent w/o syn | 67.5 | 55.2 | 43.1 | 55.0 | -25.9|
| Child w/ syn   | 68.1 | 55.7 | 43.2 | 55.4 | -25.8|
| Child w/o syn  | 67.2 | 54.4 | 43.7 | 54.2 | -25.9|
| SLAHAN w/ syn  | 69.4†| 57.6†| 45.2†| 57.3†| -23.7†|
| SLAHAN w/o syn | 67.5 | 55.2 | 43.9 | 55.0 | -25.9|

Table 6: The bold values indicate the best scores. † indicates that the difference of the score from the best baseline is statistically significant. We used paired-bootstrap-resampling with 1,000,000 random samples for the significance test (\(p < 0.05\)).