Like a word-level extractive module in the encoder-decoder model, our model can incorporate length embeddings in the decoder module for controlling the summary length. Although the length embeddings can control where to stop decoding, they do not decide which information should be included in the summary within the length constraint. Unlike the previous models, our length-controllable abstractive summarization model incorporates a word-level extractive module in the encoder-decoder model instead of length embeddings. Our model generates a summary in two steps. First, our word-level extractor extracts a sequence of important words (we call it the “prototype text”) from the source text according to the word-level importance scores and the length constraint. Second, the prototype text is used as additional input to the encoder-decoder model, which generates a summary by jointly encoding and copying words from both the prototype text and source text. Since the prototype text is a guide to both the content and length of the summary, our model can generate an informative and length-controlled summary. Experiments with the CNN/Daily Mail dataset and the NEWSROOM dataset show that our model outperformed previous models in length-controlled settings.

1 Introduction

Neural summarization has made great progress in recent years. It has two main approaches: extractive and abstractive. Extractive methods generate summaries by selecting important sentences (Zhang et al. 2018; Zhou et al. 2018). They produce grammatically correct summaries; however, they do not give much flexibility to the summarization because they only extract sentences from the source text. By contrast, abstractive summarization enables more flexible summarization, and it is expected to generate more fluent and readable summaries than extractive models. The most commonly used abstractive summarization model is the pointer-generator (See, Liu, and Manning 2017), which generates a summary word-by-word while copying words from the source text and generating words from a pre-defined vocabulary set. This model can generate an accurate summary.
by combining word-level extraction and generation.

Although the idea of controlling the length of the summary was mostly neglected in the past, it was recently pointed out that it is actually an important aspect of abstractive summarization (Liu, Luo, and Zhu 2018; Fan, Granger, and Auli 2018). In practical applications, the summary length should be controllable in order for it to fit the device that displays it. However, there have only been a few studies on controlling the summary length. Kikuchi et al. (2016) proposed a length-controllable model that uses length embeddings. In the length embedding approach, the summary length is encoded either as an embedding that represents the remaining length at each decoding step or as an initial embedding to the decoder that represents the desired length. Liu, Luo, and Zhu (2018) proposed a model that uses the desired length as an input to the initial state of the decoder. These previous models control the length in the decoding module by using length embeddings. However, length embeddings only add length information on the decoder side. Consequently, they may miss important information because it is difficult to take into account which content should be included in the summary for certain length constraints.

We propose a new length-controllable abstractive summarization that is guided by the prototype text. Our idea is to use a word-level extractive module instead of length embeddings to control the summary length. Figure 2 compares the previous length-controllable models and the proposed one. The yellow blocks are the modules responsible for length control. Since the word-level extractor controls which contents are to be included in the summary when a length constraint is given, it is possible to generate a summary including the important contents. Our model consists of two steps. First, the word-level extractor predicts the word-level importance of the source text and extracts important words according to the importance scores and the desired length. The extracted word sequence is used as a “prototype” of the summary; we call it the prototype text. Second, we use the prototype text as an additional input of the encoder-decoder model. The length of the summary is kept close to that of the prototype text because the summary is generated by referring to the prototype text. Figure 1 shows examples of output generated by our model. Our abstractive summaries are similar to the extracted prototypes. The extractive module produces a rough overview of the summary, and the encoder-decoder module produces a fluent summary based on the extracted prototype.

Our idea is inspired by extractive-and-abstractive summarization. Extractive-and-abstractive summarization incorporates an extractive model in an abstractive encoder-decoder model. While in the simple encoder-decoder model, one model identifies the important contents and generates fluent summaries, the extractive-and-abstractive model has an encoder-decoder part that generates fluent summaries and a separate part that extracts important contents. Several studies have shown that separating the problem of finding the important content and the problem of generating fluent summaries improves the accuracy of the summary (Gehrmann, Deng, and Rush 2018; Chen and Bansal 2018). Our model can be regarded as an extension of models that work in this way: However, this is the first to extend the extractive module such that it can control the summary length.

Ours is the first method that controls the summary length using an extractive module and that achieves both high accuracy and length controllability in abstractive summarization. Our contributions are summarized as follows:

- We propose a new length-controllable prototype-guided abstractive summarization model, called LPAS (Length-controllable Prototype-guided Abstractive Summarization). Our model effectively guides the abstractive summarization using a summary prototype. Our model controls the summary length by controlling the number of words in the prototype text.
- Our model achieved state-of-the-art ROUGE scores in length-controlled abstractive summarization settings on the CNN/DM and NEWSROOM datasets.

2 Task Definition

Our study defines length-controllable abstractive summarization as two pipelined tasks: prototype extraction and prototype-guided abstractive summarization. The problem formulations of each task are described below.

Task 1 (Prototype Extraction) Given a source text \( X^C \) with \( L \) words \( X^C = (x^C_1, \ldots, x^C_L) \) and a desired summary length \( K \), the model estimates importance scores \( p^\text{ext} = (p^\text{ext}_1, \ldots, p^\text{ext}_L) \) and extracts the top-\( K \) important words \( X^P = (x^P_1, \ldots, x^P_K) \) as a prototype text on the basis of \( p^\text{ext} \). The desired summary length \( K \) can be set to an arbitrary value. Note that the original word order is preserved in \( X^P \) (\( X^P \) is not bag-of-words).

Task 2 (Prototype-guided Abstractive Summarization) Given the source text and the extracted prototype text \( X^P \), the model generates a length-controlled abstractive summary \( Y = (y_1, \ldots, y_T) \). The length of summary \( T \) is controlled in accordance with the prototype length \( K \).

3 Proposed Model

3.1 Overview

Our model consists of three modules: the prototype extractor, joint encoder, and summary decoder (Figure 3). The last two modules comprise Task 2, the prototype-guided abstractive summarization. The prototype extractor uses BERT,
while the joint encoder and summary decoder use the Transformer architecture (Vaswani et al. 2017).

**Prototype extractor** (§3.2) The prototype extractor extracts the top-\(K\) important words from the source text.

**Joint encoder** (§3.3) The joint encoder encodes both the source text and the prototype text.

**Summary decoder** (§3.4) The summary decoder is based on the pointer-generator model and generates an abstractive summary by using the output of the joint encoder.

### 3.2 Prototype Extractor

Since our model extracts the prototype at the word level, the prototype extractor estimates an importance score \(p^\text{ext}_i\) of each word \(x^C_i \in X^C\). BERT has achieved SOTA on many classification tasks, so it is a natural choice for the prototype extractor. Our model uses BERT and a task-specific feed-forward network on top of BERT. We tokenize the source text using the BERT tokenizer\(^1\) and fine-tune the BERT model. The importance score \(p^\text{ext}_i\) is defined as

\[
p^\text{ext}_i = \sigma(W_1^\top \text{BERT}(X^C)_i + b_1)
\]

where \(\text{BERT()}\) is the last hidden state of the pre-trained BERT, \(W_1 \in \mathbb{R}^{d_{\text{ext}} \times d_e}\) and \(b_1\) are learnable parameters, \(\sigma\) is a sigmoid function, \(d_{\text{ext}}\) is the dimension of the last hidden state of the pre-trained BERT.

To extract a more fluent prototype than when using only the word-level importance, we define a new weighted importance score \(p^\text{ext-w}_i\) that incorporates a sentence-level importance score as a weight for the word-level importance score:

\[
p^\text{ext-w}_i = p^\text{ext}_i \cdot p^\text{ext-s}_i, \quad p^\text{ext-s}_i = \frac{1}{N_{S_j}} \sum_{l:x_l \in S_j} p^\text{ext}_l
\]

where \(N_{S_j}\) is the number of words in the \(j\)-th sentence \(S_j \in X^C\). Our model extracts the top-\(K\) important words as a prototype from the source text on the basis of \(p^\text{ext-w}_i\). It controls the length of the summary in accordance with the number of words in the prototype text, \(K\). \(1https://github.com/google-research/bert/

### 3.3 Joint Encoder

**Embedding layer** This layer projects each of the one-hot vectors of words \(x^C_i\) (of size \(V\)) into a \(d_{\text{word}}\)-dimensional vector space with a pre-trained weight matrix \(W^e \in \mathbb{R}^{d_{\text{word}} \times V}\) such as GloVe (Pennington, Socher, and Manning 2014). Then, the word embeddings are mapped to \(d_{\text{model}}\)-dimensional vectors by using the fully connected layer, and the mapped embeddings are passed to a ReLU function. This layer also adds positional encoding to the word embedding (Vaswani et al. 2017).

**Transformer encoder blocks** The encoder encodes the embedded source and prototype texts with a stack of Transformer blocks (Vaswani et al. 2017). Our model encodes the two texts with the encoder stack independently. We denote these outputs as \(E^C_s \in \mathbb{R}^{d_{\text{model}} \times L}\) and \(E^P_s \in \mathbb{R}^{d_{\text{model}} \times K}\), respectively.

**Transformer dual encoder blocks** This block calculates the interactive alignment between the encoded source and prototype texts. Specifically, it encodes the source and prototype texts and then performs multi-head attention on the other output of the encoder stack (i.e., \(E^C_s\) and \(E^P_s\)). We denote the outputs of the dual encoder stack of the source text and prototype text by \(M^C_s \in \mathbb{R}^{d_{\text{model}} \times L}\) and \(M^P_s \in \mathbb{R}^{d_{\text{model}} \times K}\), respectively.

### 3.4 Summary Decoder

**Embedding layer** The decoder receives a sequence of words in an abstractive summary \(Y\), which is generated through an auto-regressive process. At each decoding step \(t\), this layer projects each of the one-hot vectors of the words \(y_t\) in the same way as the embedding layer in the joint encoder.

**Transformer decoder blocks** The decoder uses a stack of decoder Transformer blocks (Vaswani et al. 2017) that perform multi-head attention on the encoded representations of the prototype, \(M^P_s\). It uses another stack of decoder Transformer blocks that perform multi-head attention on those of the source text, \(M^C_s\), on top of the first stack. The first stack rewrites the prototype text, and the second one complements the rewritten prototype with the original source information. The subsequent mask is used in the stacks since this component is used in a step-by-step manner at test time. The output of the stacks is \(M^T \in \mathbb{R}^{d_{\text{model}} \times T}\).

**Copying mechanism** Our pointer-generator model copies the words from the source and prototype texts on the basis of the copy distributions, for efficient reuse.

**Copy distributions** The copy distributions of the source and prototype words are described as follows:

\[
p^c(\hat{y}_t) = \sum_{k:z^C_k = \hat{y}_t} \alpha^C_{tk}, \quad p^c(\hat{y}_t) = \sum_{l:z^P_l = \hat{y}_t} \alpha^P_{tl}
\]

where \(\alpha^C_{tk}\) and \(\alpha^P_{tl}\) are respectively the first attention heads of the last block in the first and second stacks of the decoder.
The final vocabulary distribution is described as follows:

\[
p(y_t) = \lambda_g p_g(y_t) + \lambda_c p_c(y_t) + \lambda_p p_p(y_t)
\]

where \(\lambda_g\), \(\lambda_c\), and \(\lambda_p\) are learnable parameters. Also, \(p_g(y_t) = \text{softmax}(W^g(M_t^S;c_t^C;c_t^P) + b^g)\), \(c_t^C = \sum_i \alpha_{ti} M_t^C\), \(c_t^P = \sum_k \alpha_{tk} M_t^P\), and \(p_p(y_t) = \text{softmax}(W^p(M_t^S) + b^p)\) where \(W^g \in \mathbb{R}^{3 \times 3d_{model}}\), \(b^g \in \mathbb{R}^3\), \(W^p \in \mathbb{R}^{d_{model} \times V}\), and \(b^p \in \mathbb{R}^V\) are learnable parameters.

### Joint encoder and summary decoder

The main loss for the encoder-decoder is the cross-entropy loss:

\[
L_{\text{gen}} = -\frac{1}{NT} \sum_{n=1}^{N} \sum_{t=1}^{T} \log p(y_t|y_{1:t-1}, X^C, X^P).
\]

Moreover, we add the attention guide loss of the summary decoder. This loss is designed to guide the estimated attention distribution to the reference attention.

\[
L_{\text{attn}} = -\frac{1}{NT} \sum_{n=1}^{N} \sum_{t=1}^{T} \log \alpha_{t,l(t)}\]

\[
L_{\text{proto}} = -\frac{1}{NT} \sum_{n=1}^{N} \sum_{t=1}^{T} \log \alpha_{t,l(t)}^{\text{proto}}
\]

\(\alpha_{t,l(t)}\) is the first head of the last block in the joint encoder stack for the prototype. \(l(t)\) denotes the absolute position in the source text corresponding to the \(t\)-th word in the sequence of summary words. The overall loss of the generation model is a linear combination of these three losses:

\[
L_{\text{gen}} = \lambda_1 L_{\text{main}} + \lambda_2 L_{\text{attn}} + \lambda_3 L_{\text{proto}}
\]

\(\lambda_1\) and \(\lambda_2\) were set to 0.5 in the experiments.

### 5 Inference

During the inference period, we use a beam search and re-ranking (Chen and Bansal 2018). We keep all \(N_{\text{beam}}\) summary candidates provided by the beam search, where \(N_{\text{beam}}\) is the size of the beam, and generate the \(N_{\text{beam}}\)-best summaries. The summaries are then re-ranked by the number of repeated N-grams, the smaller the better. The beam search and this re-ranking improve the ROUGE score of the output, as they eliminate candidates that contain repetitions. For the length-controlled setting, we set the value of \(K\) to the desired length. For the standard setting, we set it to the average length of the reference summary in the validation data.

### 6 Experiments

#### 6.1 Datasets and settings

**Dataset** We used the CNN/DM dataset (Hermann et al. 2015), a standard corpus for news summarization. The summaries are bullet points for the articles shown on their respective websites. Following See, Liu, and Manning (2017), we used the non-anonymized version of the corpus and truncated the source documents to 400 tokens and the target summaries to 120 tokens. The dataset includes 286,817 training pairs, 13,368 validation pairs, and 11,487 test pairs. We also used the NEWSROOM dataset (Grusky, Naman, and Artzi 2018). NEWSROOM contains various news sources (38 different news sites). We used 973,042 pairs of data for training. We sampled 30,000 pairs for validation data, and the number of the test pairs was 106,349. To evaluate the length-controlled setting for NEWSROOM dataset, we randomly sampled 10,000 samples from the test set.
Table 1: ROUGE scores (F1) of abstractive summarization models with different lengths on the CNN/DM dataset (10, 30, 50, 70, 90 words). AVG indicates the average ROUGE score for the five different lengths. 1(Liu, Luo, and Zhu 2018)

| Length | Model | R-1 | R-2 | R-L |
|--------|-------|-----|-----|-----|
| 10     | LC1   | 19.03 | 8.45 | 16.47 |
|        | LenEmb | 18.19 | 8.96 | 17.44 |
|        | LPAS  | 17.43 | 8.87 | 16.78 |
| 30     | LC    | 32.26 | 13.60 | 24.64 |
|        | LenEmb | 34.01 | 15.51 | 31.43 |
|        | LPAS  | 35.11 | 17.21 | 32.83 |
| 50     | LC    | 34.74 | 14.24 | 25.62 |
|        | LenEmb | 38.66 | 17.17 | 35.49 |
|        | LPAS  | 41.47 | 19.70 | 38.46 |
| 70     | LC    | 33.83 | 13.67 | 24.67 |
|        | LenEmb | 39.57 | 17.38 | 36.22 |
|        | LPAS  | 42.48 | 19.97 | 39.25 |
| 90     | LC    | 32.17 | 13.00 | 23.28 |
|        | LenEmb | 38.51 | 16.79 | 35.24 |
|        | LPAS  | 41.54 | 19.43 | 38.30 |
| AVG    | LC    | 30.40 | 12.59 | 22.94 |
|        | LenEmb | 33.79 | 15.16 | 31.16 |
|        | LPAS  | 35.60 | 17.04 | 33.12 |

Table 1 shows that our model achieved high ROUGE scores for different lengths and outperformed the previous length-controllable models in most cases. Our model was about 2 points more accurate on average than LenEmb. Our model selected the most important words from the source text in accordance with the desired length. It was thus effective at keeping the important information even in the length-controlled setting. Figure 4a shows the precision, recall, and F score of ROUGE for different lengths. Our model maintained a high F-score around the average length (around 60 words); this indicates that it can select important information and generate stable results with different lengths.

Does our model generate a summary with the desired length? Figure 4b shows the relationship between the desired length and the output length. The x-axis indicates the desired length, and the y-axis indicates the average length and standard deviation of the length-controlled output summary. The results show that our model properly controls the summary length. This controllable nature comes from the training procedure. When training our encoder-decoder, we set the number of words $K$ in the prototype text according to the length of the reference summary; therefore, the model learns to generate a summary that has a similar length to the prototype text.

How good is the quality of the prototype text? To evaluate the quality of the prototype, we evaluated the ROUGE scores of the extracted prototype text. Table 2 shows the results. In the table, LPAS-ext (top-3 sents) means the top-three sentences were extracted using $p^\text{ext}$ψ. Interestingly, ROUGE-1 and ROUGE-2 scores of the LPAS-ext (Top-$K$ words) were higher than those of the sentence-level extractive models. This indicates that word-level LPAS-ext is effective at finding not only important words (ROUGE-1), but also important phrases (ROUGE-2). Also, we can see from Table 5 that whole LPAS improved the ROUGE-L score of LPAS-ext. This indicates that our joint encoder and summary decoder generate more fluent summaries with the help of the prototype text.

Does our abstractive model improve if the quality of the prototype is improved? We evaluated our model in the following two settings in order to analyze the relationship between the quality of the abstractive summary and that of the prototype. In the gold-length setting, we only gave the gold length $K$ to the prototype extractor. In the gold sen-

---

Model Configurations We used the same configurations for the two datasets. The extractor used the pre-trained BERT$_{large}$ model (Devlin et al. 2018). We fine-tuned BERT for two epochs with the default settings. Our encoder and decoder used pre-trained 300-dimensional GloVe embeddings. The encoder and decoder Transformer have four blocks. The number of heads was 8, and the number of dimensions of FFN was 2048. $d_{\text{model}}$ was set to 512. We used the Adam optimizer (Kingma and Ba 2015) with a scheduled learning rate (Vaswani et al. 2017). We set the size of the input vocabulary to 100,000 and the output vocabulary to 1,000.

6.2 Evaluation Metrics

We used the ROUGE scores (F1), including ROUGE-1 (R-1), ROUGE-2 (R-2), and ROUGE-L (R-L), as the evaluation metrics (Lin 2004). We used the files2rouge toolkit for calculating the ROUGE scores.

6.3 Results

Does our model improve the ROUGE score in the length-controlled setting? We used two types of length-controllable models as baselines. The first one is a CNN-based length-controllable model (LC) that uses the desired length as an input to the initial state of the CNN-based decoder. (Liu, Luo, and Zhu 2018). The second one (LenEmb) embeds the remaining length and adds them to each decoder step (Kikuchi et al. 2016). Since there are no previous results on applying LenEmb to the CNN/DM dataset, we implemented it as a Transformer-based encoder-decoder model. Specifically, we simply added the embeddings of the remaining length to the word embeddings at each decoding step.

---

https://github.com/pltrdy/files2rouge
|        | R-1 | R-2 | R-L |
|--------|-----|-----|-----|
| Lead3  | 40.3| 17.7| 36.6|
| Bottom-Up (top-3 sents) | 40.7 | 18.0 | 37.0 |
| Bottom-Up (word) | 42.0 | 15.9 | 37.3 |
| NeuSum² | 41.6 | 19.0 | 38.0 |
| BertSum³ | 43.25 | 20.24 | **39.63** |
| HIBERT⁴ | 42.37 | 19.95 | **38.83** |
| LPAS-ext |       |     |     |
| - top-3 sents | 41.48 | 19.23 | 37.76 |
| - Top-K words | **44.79** | **20.59** | 38.12 |
| AVG |       |     |     |

Table 2: ROUGE scores (F1) of our prototype extractor (LPAS-ext) on CNN/DM. ¹(Gehrmann, Deng, and Rush 2018); ²(Zhou et al. 2018); ³(Liu 2019); ⁴(Zhang, Wei, and Zhou 2019)

|        | R-1 | R-2 | R-L |
|--------|-----|-----|-----|
| Average length | 42.55 | 20.09 | 39.36 |
| Gold length | 43.23 | 20.46 | 40.00 |
| Gold sentences + Gold length | 46.68 | 23.52 | 43.41 |

Table 3: ROUGE scores (F1) of abstractive summarization models with gold settings on the CNN/DM dataset.

...the current state-of-the-art models use pre-trained encoder-decoder models (8-11), while the encoder and decoder of our model (except for prototype extractor) were not pre-trained.

We also examined the results of generating a summary from only the prototype (LPAS w/o Source) or the source sentences + the gold-length setting, we gave the gold sentences *S* oracle and gold length (see 4.1). Table 3 shows the results. These results indicate that selecting the correct number of words in the prototype improves the ROUGE scores. In this study, we simply selected the average length when extracting the prototype for all examples in the standard setting; however, there will be an improvement if we adaptively select the number of words in the prototype for each source text. Moreover, the ROUGE score largely improved in the gold sentence and gold-length settings. This indicates that the quality of the generated summary will significantly improve by increasing the accuracy of the extractive model.

**Is our model effective on other datasets?** To verify the effectiveness of our model on various other summary styles, we evaluated it on a large and varied news summary dataset, NEWSROOM. Table 4 and Figure 5 show the results in the length-controlled setting for NEWSROOM. Our model achieved higher ROUGE scores than those of LenEmb. From Figure 5a, we can see that the F-value of the ROUGE score was highest around 30 words. This is because the average word number is about 30 words. Moreover, Figure 5b shows that our model also acquired a length control capability for a dataset with various styles.

**How well does our model perform in the standard setting?** Table 5 shows that our model achieved the ROUGE scores comparable to previous models that do not consider the length constraint on the CNN/DM dataset. We note that the current state-of-the-art models use pre-trained encoder-decoder models (8-11), while the encoder and decoder of our model (except for prototype extractor) were not pre-trained.

Table 4: ROUGE scores (F1) of abstractive summarization models with different lengths on the NEWSROOM dataset.

| Length | Model     | R-1     | R-2     | R-L     |
|--------|-----------|---------|---------|---------|
| 10     | LenEmb    | **22.99** | 13.42  | 21.45  |
|        | LPAS      | 22.80   | 13.91   | **21.59** |
| 30     | LenEmb    | 37.49   | 25.67   | 34.26  |
|        | LPAS      | **39.22** | **27.33** | **35.95** |
| 50     | LenEmb    | 36.91   | 25.30   | 33.86  |
|        | LPAS      | **38.57** | **27.07** | **35.44** |
| 70     | LenEmb    | 33.52   | 23.02   | 30.90  |
|        | LPAS      | **35.29** | **24.72** | **32.62** |
| 90     | LenEmb    | 30.04   | 20.49   | 27.80  |
|        | LPAS      | **31.53** | **22.03** | **29.30** |
| AVG    | LenEmb    | **32.19** | **21.62** | **29.66** |
|        | LPAS      | **33.48** | **23.01** | **30.98** |

Figure 5: Results in the length-controlled setting on NEWSROOM. a): ROUGE-L recall, precision and F scores for different lengths (left). b): Output length distribution (right).

(LPAS w/o Prototype). Here, using only the prototype, turned out to have the same accuracy as using only the source, but the model using the source and the prototype simultaneously had higher accuracy. These results indicate that our prototype extraction and joint encoder effectively incorporated the source text and prototype information and contributed to improving the accuracy.

The results for the NEWSROOM dataset under standard settings are shown in Table 6. To consider differences in summary length between news domains, we evaluated our model in the average length and domain-level average length (denoted as domain length) settings. The results indicate that our model had significantly higher ROUGE scores compared with the official baselines and outperformed our baseline (LPAS w/o Prototype). They also indicate that our model is effective on datasets containing text in various styles. Moreover, we found that considering the domain length has positive effects on the ROUGE scores. This indicates that our model can easily reflect differences in summary length among various styles.

**7 Related Work and Discussion**

**Length control for summarization** Kikuchi et al. (2016) were the first to propose using length embedding for length-controlled abstractive summarization. Fan, Grangier, and Auli (2018) also used length embeddings at the beginning of the decoder module for length control. Liu, Luo, and Zhu (2018) proposed a CNN-based length-controllable summarization model that uses the desired length as an in-
Table 5: ROUGE scores (F1) of abstractive summarization models on CNN/DM. 1(See, Liu, and Manning 2017); 2(Li et al. 2018); 3(Hsu et al. 2018); 4(Chen and Bansal 2018); 5(Gehrmann, Deng, and Rush 2018); 6(Mendes et al. 2019); 7(You et al. 2019); 8(Wang et al. 2019); 9(Dong et al. 2019); 10(Raffel et al. 2019); 11(Lewis et al. 2019). LPAS w/o Prototype denotes a simple Transformer-based pointer-generator, which is our model without the prototype extractor and the joint encoder. LPAS w/o Source denotes a model that generates a summary only from the prototype text.

| Model                      | R-1   | R-2   | R-L   |
|----------------------------|-------|-------|-------|
| w/o pre-trained encoder-decoder model |       |       |       |
| Pointer-Generator$^1$      | 36.44 | 15.66 | 33.42 |
| Pointer-Generator + Coverage$^1$ | 39.53 | 17.28 | 36.38 |
| Key information guide network$^2$ | 38.95 | 17.12 | 35.68 |
| Uniform summarization$^3$  | 40.68 | 17.97 | 37.13 |
| Sentence-rewriting$^4$     | 40.88 | 17.80 | 38.54 |
| Bottom-Up$^5$              | 41.22 | 18.68 | 38.34 |
| EXCONSUMM Compressive$^6$  | 40.9  | 18.0  | 37.4  |
| ETADS$^7$                  | 41.75 | 19.01 | 38.89 |
| LPAS w/o Prototype         | 42.55 | 20.09 | 39.36 |
| LPAS w/o Source            | 40.71 | 18.43 | 37.32 |

| Model                      | R-1   | R-2   | R-L   |
|----------------------------|-------|-------|-------|
| w/ pre-trained encoder-decoder model |       |       |       |
| PoDA$^8$                   | 41.87 | 19.27 | 38.54 |
| UniLM$^9$                  | 43.47 | 20.30 | 40.63 |
| T5$^{10}$                  | 43.52 | 21.50 | 40.69 |
| BART$^{11}$                | 44.16 | 21.28 | 40.90 |

Table 6: ROUGE scores (F1) of proposed models on NEWSROOM dataset. 1(Grusky, Naaman, and Artzi 2018)

| Model                      | R-1   | R-2   | R-L   |
|----------------------------|-------|-------|-------|
| Lead3$^3$                  | 32.02 | 21.08 | 29.59 |
| pointer-generator$^4$      | 27.54 | 13.32 | 23.50 |
| LPAS$^8$                   | 39.24 | 27.20 | 35.84 |
| $K = \text{average length}$ | 39.79 | 27.85 | 36.48 |
| $K = \text{domain length}$ | 38.48 | 26.99 | 35.30 |

Neural extractive-and-abstractive summarization

Hsu et al. (2018), Gehrmann, Deng, and Rush (2018) and You et al. (2019) incorporated a sentence- and word-level extractive model in the pointer-generator model. Their models weight the copy probability for the source text by using an extractive model and guide the pointer-generator model to copy important words. Li et al. (2018) proposed a keyword-guided abstractive summarization model. Chen and Bansal (2018) proposed a sentence extraction and re-writing model that trains in an end-to-end manner by using reinforcement learning. Cao et al. (2018) proposed a search and rewrite model. Mendes et al. (2019) proposed a combination of sentence-level extraction and compression. The idea behind these models is word-level weighting for the entire source text or sentence-level re-writing. On the other hand, our model guides the summarization with a length-controllable prototype text by using the prototype extractor and joint encoder. Utilizing extractive results to control the length of the summary is a new idea.

Large-scale pre-trained language model

BERT (Devlin et al. 2018) is a new pre-trained language model that uses bidirectional encoder representations from Transformer. BERT has performed well in many natural language understanding tasks such as the GLUE benchmarks (Wang et al. 2018) and natural language inference (Williams, Nangia, and Bowman 2018). Liu (2019) used BERT for their sentence-level extractive summarization model. Zhang, Wei, and Zhou (2019) trained a new pre-trained model that considers document-level information for sentence-level extractive summarization. We used BERT for the word-level prototype extractor and verified the effectiveness of using it in the word-level extractive module. Several researchers have published pre-trained encoder-decoder models very recently (Wang et al. 2019; Lewis et al. 2019; Raffel et al. 2019). Wang et al. (2019) pre-trained a transformer-based pointer-generator model. Lewis et al. (2019) pre-trained a normal transformer-based encoder-decoder model using large unlabeled data and achieved state-of-the-art results. Dong et al. (2019) extended the BERT structure to handle sequence-to-sequence tasks.

Reinforcement learning for summarization

Reinforcement learning (RL) is a key summarization technique. RL can be used to optimize non-differential metrics or multiple non-differential networks. Narayan, Cohen, and Lapata (2018) and Dong et al. (2018) used RL for extractive summarization. For abstractive summarization, Paulus, Xiong, and Socher (2017) used RL to mitigate the exposure bias of abstractive summarization. Chen and Bansal (2018) used RL to combine sentence-extraction and pointer-generator models. Our model achieved high ROUGE scores without RL. In future, we may incorporate RL in our models to get a further improvement.

8 Conclusion

We proposed a new length-controllable abstractive summarization model. Our model consists of a word-level prototype extractor and a prototype-guided abstractive summarization model. The prototype extractor identifies the important part of the source text within the length constraint, and the abstractive model is guided with the prototype text. This characteristic enabled it to achieve a high ROUGE score in standard summarization tasks. Moreover, our prototype extractor ensures the summary will have the desired length. Experiments with the CNN/DM dataset and the NEWSROOM dataset show that our model outperformed previous models in standard and length-controlled settings. In future, we put to the initial state of the decoder. Takase and Okazaki (2019) introduced positional encoding that represents the remaining length at each decoder step of the Transformer-based encoder-decoder model. It is almost equivalent to the model LenEmb we implemented. These previous models use length embeddings for controlling the length in the decoding module, whereas we use the prototype extractor for controlling the summary length and to include important information in the summary.

Neural extractive-and-abstractive summarization

Hsu et al. (2018), Gehrmann, Deng, and Rush (2018) and You et al. (2019) incorporated a sentence- and word-level extractive model in the pointer-generator model. Their models weight the copy probability for the source text by using an extractive model and guide the pointer-generator model to copy important words. Li et al. (2018) proposed a keyword-guided abstractive summarization model. Chen and Bansal (2018)
would like to incorporate a pre-trained language model in the abstractive model to build a higher quality summarization model.

References
Cao, Z.; Li, W.; Li, S.; and Wei, F. 2018. Retrieve, rerank and rewrite: Soft template based neural summarization. In ACL, 152–161.

Chen, Y.-C., and Bansal, M. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In EMNLP, 675–686.

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

Dong, Y.; Shen, Y.; Crawford, E.; van Hoof, H.; and Cheung, J. C. K. 2018. BanditSum: Extractive summarization as a contextual bandit. In EMNLP, 3739–3748.

Dong, L.; Yang, N.; Wang, W.; Wei, F.; Liu, X.; Wang, Y.; Gao, J.; Zhou, M.; and Hon, H.-W. 2019. Unified language model pre-training for natural language understanding and generation. In Advances in Neural Information Processing Systems 32, 13042–13054.

Fan, A.; Grangier, D.; and Auli, M. 2018. Controllable abstractive summarization. In NMT@ACL, 45–54.

Gehrmann, S.; Deng, Y.; and Rush, A. 2018. Bottom-up abstractive summarization. In EMNLP, 4098–4109.

Grusky, M.; Naaman, M.; and Artzi, Y. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In ACL, 708–719.

Hermann, K. M.; Kocisky, T.; Grefenstette, E.; Espeholt, L.; Kay, W.; Suleyman, M.; and Blunsom, P. 2015. Teaching machines to read and comprehend. In NIPS, 1693–1701.

Hsu, W.-T.; Lin, C.-K.; Lee, M.-Y.; Min, K.; Tang, J.; and Sun, M. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. In ACL, 132–141.

Kikuchi, Y.; Neubig, G.; Sasano, R.; Takamura, H.; and Okumura, M. 2016. Controlling output length in neural encoder-decoders. In EMNLP, 1328–1338.

Kingma, D. P., and Ba, J. 2015. Adam: A method for stochastic optimization. In ICLR.

Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv e-prints.

Li, C.; Xu, W.; Li, S.; and Gao, S. 2018. Guiding generation for abstractive text summarization based on key information guide network. In ACL, 55–60.

Lin, C.-Y. 2004. Rouge: A package for automatic evaluation of summaries. In ACL.

Liu, Y.; Luo, Z.; and Zhu, K. 2018. Controlling length in abstractive summarization using a convolutional neural network. In EMNLP, 4110–4119.

Liu, Y. 2019. Fine-tune BERT for extractive summarization. CoRR abs/1903.10318.

Mendes, A.; Narayan, S.; Miranda, S.; Marinho, Z.; Martins, A. F. T.; and Cohen, S. B. 2019. Jointly extracting and compressing documents with summary state representations. In NAACL, 3955–3966.

Narayan, S.; Cohen, S. B.; and Lapata, M. 2018. Ranking sentences for extractive summarization with reinforcement learning. In NAACL, 1747–1759.

Paulus, R.; Xiong, C.; and Socher, R. 2017. A deep reinforced model for abstractive summarization. CoRR abs/1705.04304.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In EMNLP.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv e-prints.

See, A.; Liu, P. J.; and Manning, C. D. 2017. Get to the point: Summarization with pointer-generator networks. In ACL, 1073–1083.

Takase, S., and Okazaki, N. 2019. Positional encoding to control output sequence length. In NAACL, 3999–4004.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is all you need. In NIPS, 5998–6008.

Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. R. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In BlackboxNLP@EMNLP, 353–355.

Wang, L.; Zhao, W.; Jia, R.; Li, S.; and Liu, J. 2019. Denosing based sequence-to-sequence pre-training for text generation. In EMNLP, to appear.

Williams, A.; Nangia, N.; and Bowman, S. R. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In NAACL-HLT, 1112–1122.

You, Y.; Jia, W.; Liu, T.; and Yang, W. 2019. Improving abstractive document summarization with salient information modeling. In ACL, 2132–2141.

Zhang, X.; Lapata, M.; Wei, F.; and Zhou, M. 2018. Neural latent extractive document summarization. In EMNLP, 779–784. Association for Computational Linguistics.

Zhang, X.; Wei, F.; and Zhou, M. 2019. HIBERT: document level pre-training of hierarchical bidirectional transformers for document summarization. In ACL, 5059–5069.

Zhou, Q.; Yang, N.; Wei, F.; Huang, S.; Zhou, M.; and Zhao, T. 2018. Neural document summarization by jointly learning to score and select sentences. In ACL, 654–663.