How to Write Summaries with Patterns?
Learning towards Abstractive Summarization through Prototype Editing

Shen Gao ♠*, **Xiuying Chen ♠♣***, **Piji Li ♢**, **Zhangming Chan ♠♣**, **Dongyan Zhao ♠♣**, **Rui Yan ♠♣†**

♠Wangxuan Institute of Computer Technology, Peking University, Beijing, China,
♢Tencent AI Lab, Shenzhen, China
♣Center for Data Science, Peking University, Beijing, China
{shengao, xy-chen, zhangming.chan, zhaody, ruiyan}@pku.edu.cn
pijili@tencent.com

Abstract

Under special circumstances, summaries should conform to a particular style with patterns, such as court judgments and abstracts in academic papers. To this end, the prototype document-summary pairs can be utilized to generate better summaries. There are two main challenges in this task: (1) the model needs to incorporate learned patterns from the prototype, but (2) should avoid copying contents other than the patternized words— such as irrelevant facts—into the generated summaries. To tackle these challenges, we design a model named *Prototype Editing based Summary Generator* (PESG). PESG first **learns summary patterns** and prototype facts by analyzing the correlation between a prototype document and its summary. Prototype facts are then utilized to help **extract facts** from the input document. Next, an editing generator generates new summary based on the **summary pattern** or **extracted facts**. Finally, to address the second challenge, a fact checker is used to estimate mutual information between the input document and generated summary, providing an additional signal for the generator. Extensive experiments conducted on a large-scale real-world text summarization dataset[1] show that PESG achieves the state-of-the-art performance in terms of both automatic metrics and human evaluations.

1 Introduction

Abstractive summarization can be regarded as a sequence mapping task that maps the source text to the target summary (Rush et al., 2015; Li et al., 2017; Cao et al., 2018; Gao et al., 2019a). It has drawn significant attention since the introduction

---

*Equal contribution. Ordering is decided by a coin flip.
†Corresponding author.
[1] https://github.com/gsh199449/proto-summ

Table 1: An example of patternized summary generation. The text in red denotes patternized words shared in different summaries, and text in blue denotes specific facts.

of deep neural networks to natural language processing. Under special circumstances, the generated summaries are required to conform to a specific pattern, such as court judgments, diagnosis certificates, abstracts in academic papers, etc. Take the court judgments for example, there is always a statement of the crime committed by the accused, followed by the motives and the results of the judgment. An example case is shown in Table 1, where the summary shares the same writing style and has words in common with the prototype summary (retrieved from the training dataset).

Existing prototype based generation models such as (Wu et al., 2018) are all applied on short text, thus, cannot handle long documents summarization task. Another series of works focus on template-based methods such as (Oya et al., 2014). However, template-based methods are too rigid for our patternized summary generation task. Hence, in this paper, we propose a summarization framework named *Prototype Editing based Summary Generator* (PESG) that incorporates prototype document-summary pairs to improve summa-

rization performance when generating summaries with pattern. First, we calculate the cross dependency between the prototype document-summary pair to obtain a summary pattern and prototype facts (explained in § 4.2). Then, we extract facts from the input document with the help of the prototype facts (explained in § 4.3). Next, a recurrent neural network (RNN) based decoder is used to generate a new summary, incorporating both the summary pattern and extracted facts (explained in § 4.4). Finally, a fact checker is designed to provide mutual information between the generated summary and the input document to prevent the generator from copying irrelevant facts from the prototype (explained in § 4.5). To evaluate PESG, we collect a large-scale court judgment dataset, where each judgment is a summary of the case description with a patternized style. Extensive experiments conducted on this dataset show that PESG outperforms the state-of-the-art summarization baselines in terms of ROUGE metrics and human evaluations by a large margin.

Our contributions can be summarized as follows:

- We propose to use prototype information to help generate better summaries with patterns.

- Specifically, we propose to generate the summary incorporating the prototype summary pattern and extracted facts from input document.

- We provide mutual information signal for the generator to prevent copying irrelevant facts from the prototype.

- We release a large-scale prototype based summarization dataset that is beneficial for the community.

2   Related Work

We detail related work on text summarization and prototype editing.

Text summarization can be classified into extractive and abstractive methods. Extractive methods (Narayan et al., 2018b; Chen et al., 2018) directly select salient sentences from an article to compose a summary. One shortcoming of these models is that they tend to suffer from redundancy. Recently, with the emergence of neural network models for text generation, a vast majority of the literature on summarization (Ma et al., 2018; Zhou et al., 2018; Gao et al., 2019a; Chen et al., 2019) is dedicated to abstractive summarization, which aims to generate new content that concisely paraphrases a document from scratch.

Another line of research focuses on prototype editing. (Guu et al., 2018) proposed the first prototype editing model, which samples a prototype sentence from training data and then edits it into a new sentence. Following this work, (Wu et al., 2018) proposed a new paradigm for response generation, which first retrieves a prototype response from a pre-defined index and then edits the prototype response. (Cao et al., 2018) applied this method on summarization, where they employed existing summaries as soft templates to generate new summary without modeling the dependency between the prototype document, summary and input document. Different from these soft attention methods, (Cai et al., 2018) proposed a hard-editing skeleton-based model to promote the coherence of generated stories. Template-based summarization is also a hard-editing method (Oya et al., 2014), where a multi-sentence fusion algorithm is extended in order to generate summary templates.

Different from all above works, our model focuses on patternized summary generation, which is more challenging than traditional news summarization and short sentence prototype editing.

3   Problem Formulation

For an input document $X = \{x_1, x_2, \ldots, x_{T_m}\}$, we assume there is a ground truth summary $Y = \{y_1, y_2, \ldots, y_{T_n}\}$. In our prototype summarization task, a retrieved prototype document $\hat{X} = \{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{T_m}\}$ with a corresponding prototype summary $\hat{Y} = \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{T_n}\}$ is also attached according to their similarities with X.

For a given document X, our model extracts salient facts from X guided by a prototype document $\hat{X}$, and then generates the summary Y' by referring to the prototype summary $\hat{Y}$. The goal is to generate a summary Y' that not only follows a patternized style (as defined by prototype summary $\hat{Y}$) but also is consistent with the facts in document X.

4   Model

4.1   Overview

In this section, we propose our *prototype editing based summary generator*, which can be split into two main parts, as shown in Figure 1:

- *Summary Generator.* (1) **Prototype Reader** analyzes the dependency between $\hat{X}$ and $\hat{Y}$ to de-
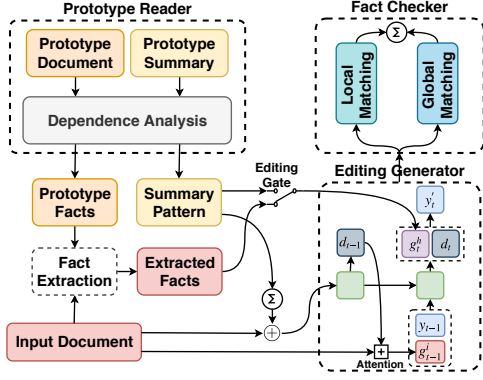
Figure 1: Overview of PESG. We divide our model into four parts: (1) *Prototype Reader*; (2) *Fact Extraction*; (3) *Editing Generator*; (4) *Fact Checker*.

termine the summary pattern and prototype facts. (2) **Fact Extraction** module extracts facts from the input document under the guidance of the prototype facts. (3) **Editing Generator** module generates the summary Y' of document X by incorporating summary pattern and facts.

• *Fact Checker* estimates the mutual information between the generated summary Y' and input document X. This information provides an additional signal for the generation process, preventing irrelevant facts from being copied from the prototype document.

4.2 Prototype Reader

To begin with, we use an embedding matrix e to map a one-hot representation of each word in X, $\hat{X}$, $\hat{Y}$ into a high-dimensional vector space. We then employ a bi-directional recurrent neural network (Bi-RNN) to model the temporal interactions between words:

$$h_t^x = \text{Bi-RNN}_x(e(x_t), h_{t-1}^x), \tag{1}$$

$$\hat{h}_t^x = \text{Bi-RNN}_x(e(\hat{x}_t), \hat{h}_{t-1}^x), \tag{2}$$

$$\hat{h}_t^y = \text{Bi-RNN}_y(e(\hat{y}_t), \hat{h}_{t-1}^y), \tag{3}$$

where h_t^x, $\hat{h}_t^x$ and $\hat{h}_t^y$ denote the hidden state of t-th step in Bi-RNN for X, $\hat{X}$ and $\hat{Y}$, respectively. Following (Tao et al., 2018; Gao et al., 2019b; Hu et al., 2019), we choose long short-term memory (LSTM) as the cell for Bi-RNN.

On one hand, the sections in the prototype summary that are not highly related to the prototype document are the universal patternized words and should be emphasized when generating the new summary. On the other hand, the sections in the prototype document that are highly related to the

prototype summary are useful facts that can guide the process of extracting facts from input document. Hence, we employ a bi-directional attention mechanism between a prototype document and summary to analyze the cross-dependency, that is, from document to summary and from summary to document. Both of these attentions are derived from a shared similarity matrix, $S \in \mathbb{R}^{T_m \times T_n}$, calculated by the hidden states of prototype document $\hat{X}$ and prototype summary $\hat{Y}$. S_{ij} indicates the similarity between the i-th document word $\hat{x}_i$ and j-th summary word $\hat{y}_j$ and is computed as:

$$S_{ij} = \alpha(\hat{h}_i^x, \hat{h}_j^y),$$
$$\alpha(x,y) = w^\mathsf{T}[x \oplus y \oplus (x \otimes y)], \tag{4}$$

where α is a trainable scalar function that calculates the similarity between two input vectors. $\oplus$ denotes a concatenation operation and $\otimes$ is an element-wise multiplication.

We use $a_t^s = \text{mean}(S_{:t}) \in \mathbb{R}$ to represent the attention weight on the t-th prototype summary word by document words, which will learn to assign high weights to highly related universal patternized words when generating a summary. From a_t^s, we obtain the weighted sum of the hidden states of prototype summary as "**summary pattern**" $l = \{l_1, \ldots, l_{T_n}\}$, where l_i is:

$$l_i = a_i^s \hat{h}_i^y. \tag{5}$$

Similarly, $a_t^d = \text{mean}(S_{t:}) \in \mathbb{R}$ assigns high weights to the words in a prototype document that are relevant to the prototype summary. A convolutional layer is then applied to extract "**prototype facts**" $\hat{r}_t$ from the prototype document:

$$\hat{r}_t = \text{CNN}(a_t^d \hat{h}_t^x). \tag{6}$$

We sum the prototype facts to obtain the overall representation of these facts:

$$q = \sum_t^{T_m} \hat{r}_t. \tag{7}$$

4.3 Fact Extraction

In this section, we discuss how to extract useful facts from an input document with the help of prototype facts.

We first extract the facts from an input document by calculating their relevance to prototype facts. The similarity matrix E is then calculated between the weighted prototype document $a_i^d \hat{h}_i^x$ and input document representation h_j^x:
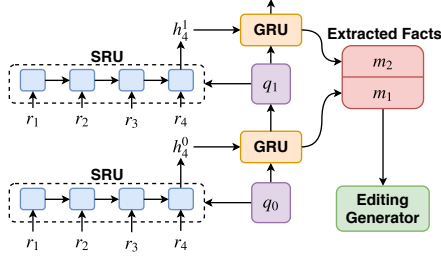
$$E_{ij} = \alpha(a_i^d \hat{h}_i^x, h_j^x), \tag{8}$$

3743

Figure 2: Framework of fact extraction module.

where α is the similarity function introduced in Equation 4. Then, we sum up E_{ij} along the length of the prototype document to obtain the weight $E_j = \sum_{ti}^{T_m} E_{tj}$ for j-th word in the document. Next, similar to Equation 6, a convolutional layer is applied on the weighted hidden states $E_t h_t^x$ to obtain the fact representation r_t from the input document:

$$r_t = \mathrm{CNN}(E_t h_t^x). \tag{9}$$

Inspired by the polishing strategy in extractive summarization (Chen et al., 2018), we propose to use the prototype facts to polish the extracted facts r_t and obtain the final fact representation $m_.$, as shown in Figure 2. Generally, the polishing process consists of two hierarchical recurrent layers. The first recurrent layer is made up of Selective Recurrent Units (SRUs), which take facts $r_.$ and polished fact q_k as input, outputting the hidden state $h_{T_m}^k$. The second recurrent layer consists of regular Gated Recurrent Units (GRUs), which are used to update the polished fact from q_k to q_{k+1} using $h_{T_m}^k$.

SRU is a modified version of the original GRU introduced in (Chen et al., 2018), details of which can be found in Appendix A. Its difference from GRU lies in that the update gate in SRU is decided by both the polished fact q_k and original fact r_t together. The t-th hidden state of SRU is calculated as:

$$h_t^k = \mathrm{SRU}(r_t, q_k). \tag{10}$$

We take $h_{T_m}^k$ as the overall representation of all input facts $r_.$. In this way, SRU can decide to which degree each unit should be updated based on its relationship with the polished fact q_k.

Next, $h_{T_m}^k$ is used to update the polished fact q_k using the second recurrent layer, consisting of GRUs:

$$m_{k+1}, q_{k+1} = \mathrm{GRU}(h_{T_m}^k, q_k), \tag{11}$$

where q_k is the cell state, $h_{T_m}^k$ is the input and m_{k+1} is the output hidden state. q_0 is initialized using q in Equation 7. This iterative process is conducted K times, and each output m_k is stored as **extracted facts** $M = \{m_1, m_2, \ldots, m_K\}$. In this way, M stores facts with different polished levels.

4.4 Editing Generator

The editing generator aims to generate a summary based on the input document, prototype summary and extracted facts. As with the settings of prototype reader, we use LSTM as the RNN cell. We first apply a linear transformation on the summation of the summary pattern $l' = \sum_i^{T_n} l_i$ and input document representations $h_{T_m}^x$, and then employ this vector as the initial state d_0 of the RNN generator as shown in Equation 12. The procedure of t-th generation is shown in Equation 13:

$$d_0 = W_e[h_{T_m}^x \oplus l'] + b_e, \tag{12}$$
$$d_t = \mathrm{LSTM}(d_{t-1}, [g_{t-1}^i \oplus e(y_{t-1})]), \tag{13}$$

where W_e, b_e are trainable parameters, d_t is the hidden state of the t-th generating step, and g_{t-1}^i is the context vector produced by the standard attention mechanism (Bahdanau et al., 2015).

To take advantage of the extracted facts M and prototype summary l, we incorporate them both into summary generation using a dynamic attention. More specifically, we utilize a matching function f to model the relationship between the current decoding state d_t and each v_i (v_i can be a extracted fact m_i or summary pattern l_i):

$$\delta_{it} = \frac{\exp(f(v_i, d_t))}{\sum_j^K \exp(f(v_j, d_t))}, \tag{14}$$
$$g_t^* = \sum_i^K \delta_{it} v_i, \tag{15}$$

where g_t^* can be g_t^m or g_t^s for attending to extracted facts or a summary pattern, respectively. We use a simple but efficient bi-linear layer as the matching function $f = m_i W_f d_t$. As for combining g_t^m and g_t^s, we propose to use an "editing gate" γ, which is determined by the decoder state d_t, to decide the importance of the summary pattern and extracted facts at each decoding step.

$$\gamma = \sigma\left(W_g d_t + b_g\right), \tag{16}$$

where σ denotes the sigmoid function. Using the editing gate, we obtain g_t^h which dynamically

combines information from the extracted facts and summary pattern with the editing gate γ, as:

$$g_t^h = [\gamma g_t^m \oplus (1 - \gamma) g_t^s]. \qquad (17)$$

Finally, the context vector g_t^h is concatenated with the decoder state d_t and fed into a linear layer to obtain the generated word distribution P_v:

$$d_t^o = W_o[d_t \oplus g_t^h] + b_o, \qquad (18)$$
$$P_v = \mathrm{softmax}\left(W_v d_t^o + b_v\right). \qquad (19)$$

The loss is the negative log likelihood of the target word y_t:

$$\mathcal{L}_s = -\sum_{t=1}^{T_n} \log P_v(y_t). \qquad (20)$$

In order to handle the out-of-vocabulary (OOV) problem, we equip our decoder with a pointer network (Gu et al., 2016; Vinyals et al., 2015; See et al., 2017). This process is the same as the model described in (See et al., 2017), thus, is omit here due to limited space.

What's more, previous work (Holtzman et al., 2018) has found that using a cross entropy loss alone is not enough for generating coherent text. Similarly, in our task, using $\mathcal{L}_s$ alone is not enough to distinguish a good summary with accurate facts from a bad summary with detailed facts from the prototype document (see § 6.2). Thus, we propose a fact checker to determine whether the generated summary is highly related to the input document.

4.5 Fact Checker

To generate accurate summaries that are consistent with the detailed facts from the input document rather than facts from the prototype document, we add a fact checker to provide additional training signals for the generator. Following (Hjelm et al., 2018), we employ the neural mutual information estimator to estimate the mutual information between the generated summary Y' and its corresponding document X, as well as the prototype document $\hat{X}$. Generally, mutual information is estimated from a local and global level, and we expect the matching degree to be higher between the generated summary and input document than the prototype document. An overview of the fact checker is shown in Figure 3.

To begin, we use a local matching network to calculate the matching degree, for local features, between the generated summary and the input, as well as prototype document. Remember that, in
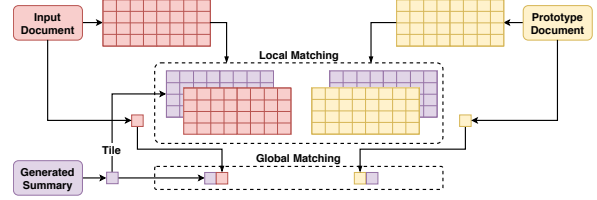

Figure 3: Framework of fact checker module.

§ 4.3, we obtain the fact representation of an input document r_i and prototype facts $\hat{r}_i$. Combining these with the final hidden state d_{T_n} of the generator RNN (in Equation 13), yields the local features of input extracted facts and the prototype facts:

$$C^r = \{d_{T_n} \oplus r_1, \ldots, d_{T_n} \oplus r_{T_m}\}, \qquad (21)$$
$$C^f = \{d_{T_n} \oplus \hat{r}_1, \ldots, d_{T_n} \oplus \hat{r}_{T_m}\}. \qquad (22)$$

A 1×1 convolutional layer and a fully-connected layer are applied to score these two features:

$$\tau_l^r = \mathrm{CNN}_l(C^r), \quad \tau_l^f = \mathrm{CNN}_l(C^f), \qquad (23)$$

where $\tau_l^r \in \mathbb{R}$, $\tau_l^f \in \mathbb{R}$ represent the local matching degree between the generated summary and input document and prototype document, respectively. We want the generated summary to be more similar to the input document than the prototype document. Thus, the optimization objective of the local matching network is to minimize $\mathcal{L}_l$:

$$\mathcal{L}_l = -\left(\log(\tau_l^r) + \log(1 - \tau_l^f)\right). \qquad (24)$$

We also have a global matching network to measure the matching degree, for global features, between the generated summary and the input document, as well as prototype document. To do so, we concatenate the representation of the generated summary with the final hidden state of the input document $h_{T_m}^x$ and final state of the prototype document $\hat{h}_{T_m}^x$, respectively, and apply a linear layer to these:

$$\tau_g^r = \mathrm{relu}(W_m[d_{T_n} \oplus h_{T_m}^x] + b_m), \qquad (25)$$
$$\tau_g^f = \mathrm{relu}(W_m[d_{T_n} \oplus \hat{h}_{T_m}^x] + b_m), \qquad (26)$$

where W_m, b_m are trainable parameters and $\tau_g^r \in \mathbb{R}$ and $\tau_g^f \in \mathbb{R}$ represent the matching degree between the generated summary and the input document, and prototype document, respectively. The objective of this global matching network, similar to the local matching network, is to minimize:

$$\mathcal{L}_g = -\left(\log(\tau_g^r) + \log(1 - \tau_g^f)\right). \qquad (27)$$

Acronym	Gloss
PESG-FC	PESG w/o Fact Checker
PESG-PR	PESG w/o Prototype Reader
PESG-SS	PESG w/o Summary Pattern
PESG-FG	PESG w/o FGRU

Table 2: Ablation models for comparison.

Finally, we combine the local and global loss functions to obtain the final loss $\mathcal{L}$, which we use $\mathcal{L}$ to calculate the gradients for all parameters:

$$\mathcal{L} = \epsilon\mathcal{L}_g + \eta\mathcal{L}_l + \mathcal{L}_s, \tag{28}$$

where ϵ, η are both hyper parameters. To optimize the trainable parameters, we employ the gradient descent method Adagrad (Duchi et al., 2010) to update all parameters.

5 Experimental Setup

5.1 Dataset

We collect a large-scale prototype based summarization dataset[2], which contains 2,003,390 court judgment documents. In this dataset, we use a case description as an input document and the court judgment as the summary. The average lengths of the input documents and summaries are 595.15 words and 273.57 words respectively. The percentage of words common to a prototype summary and the reference summary is 80.66%, which confirms the feasibility and necessity of prototype summarization. Following other summarization datasets (Grusky et al., 2018; Kim et al., 2019; Narayan et al., 2018a), we also count the novel n-grams in a summary compared with the n-grams in the original document, and the percentage of novel n-grams are 51.21%, 84.59%, 91.48%, 94.83% for novel 1-grams to 4-grams respectively. The coverage, compression and density (Grusky et al., 2018) are commonly used as metrics to evaluate the abstractness of a summary. For the summaries in our dataset, the coverage percentage is 48.78%, compression is 2.28 and density is 1.31. We anonymize entity tokens into special tags, such as using "PERS" to replace a person's name.

5.2 Comparisons

In order to prove the effectiveness of each module of PESG, we conduct several ablation studies, shown in Table 2. We also compare our model with the following baselines: (1) Lead-3 is a

---

[2]https://github.com/gsh199449/proto-summ

commonly used summarization baseline (Nallapati et al., 2017; See et al., 2017), which selects the first three sentences of document as the summary. (2) S2S is a sequence-to-sequence framework with a pointer network, proposed by (See et al., 2017). (3) Proto is a context-aware prototype editing dialog response generation model proposed by (Wu et al., 2018). (4) Re^{3}Sum, proposed by (Cao et al., 2018), uses an IR platform to retrieve proper summaries and extends the seq2seq framework to jointly conduct template-aware summary generation. (5) Uni-model was proposed by (Hsu et al., 2018), and is the current state-of-the-art abstractive summarization approach on the CNN/DailyMail dataset. (6) We also directly concatenate the prototype summary with the original document as input for S2S and Uni-model, named as Concat-S2S and Concat-Uni, respectively.

5.3 Evaluation Metrics

For the court judgment dataset, we evaluate standard ROUGE-1, ROUGE-2 and ROUGE-L (Lin, 2004) on full-length F1 following previous works (Nallapati et al., 2017; See et al., 2017; Paulus et al., 2018), where ROUGE-1 (R1), ROUGE-2 (R2), and ROUGE-L (RL) refer to the matches of unigram, bigrams, and the longest common subsequence respectively.

(Schluter, 2017) notes that only using the ROUGE metric to evaluate summarization quality can be misleading. Therefore, we also evaluate our model by human evaluation. Three highly educated participants are asked to score 100 randomly sampled summaries generated by three models: Uni-model, Re^{3}Sum and PESG. The statistical significance of observed differences between the performance of two runs is tested using a two-tailed paired t-test and is denoted using ▲(or ▼) for strong (or weak) significance for $\alpha = 0.01$.

5.4 Implementation Details

We implement our experiments in Tensor-Flow (Abadi et al., 2016) on an NVIDIA GTX 1080 Ti GPU. The word embedding dimension is 256 and the number of hidden units is 256. The batch size is set to 64. We padded or cut input document to contain exactly 250 words, and the decoding length is set to 100. ϵ and η from the Equation 28 are both set to 1.0. We initialize all of the parameters randomly using a Gaussian distribution. We use Adagrad optimizer (Duchi
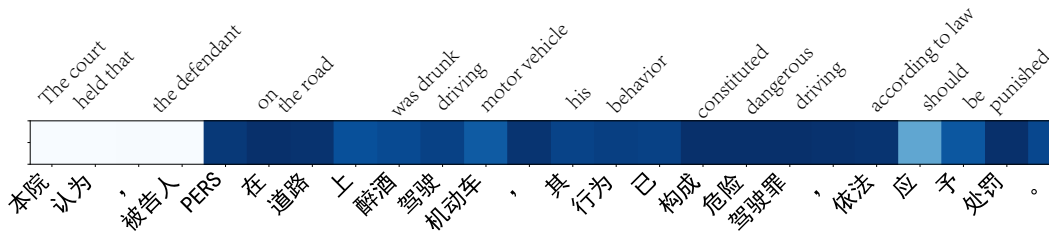
Figure 4: Visualizations of editing gate.

	R1	R2	RL
Lead-3	32.9	13.3	30.0
Re³Sum	36.3	24.0	36.0
S2S	37.6	24.6	37.3
Uni-model	37.9	25.0	37.6
Concat-S2S	34.2	20.3	34.3
Concat-Uni	37.4	24.3	36.9
PESG	**40.2**	**28.1**	**39.9**
Proto	–	–	–

Table 3: ROUGE scores comparison with baselines. `Proto` directly copies from the prototype summary as generated summary.

	Fluency	Consistency
Uni-model	1.61	1.53
Re³Sum	1.53	1.14
PESG	**1.86**▲	**1.73**▲

Table 4: Fluency and consistency comparison by human evaluation.

et al., 2010) as our optimizing algorithm and employ beam search with size 5 to generate more fluency summary sentence. We also apply gradient clipping (Pascanu et al., 2013) with range $[-5, 5]$ during training. We use dropout (Srivastava et al., 2014) as regularization with keep probability $p = 0.7$.

6 Experimental Result

6.1 Overall Performance

We compare our model with the baselines listed in Table 3. Our model performs consistently better than other summarization models including the state-of-the-art model with improvements of 6%, 12% and 6% in terms of ROUGE-1, ROUGE-2 and ROUGE-L. This demonstrates that prototype document-summary pair provides strong guidance for summary generation that cannot be replaced by other complicated baselines without prototype information. Meanwhile, directly concatenating the

prototype summary with the original input does not increase performance, instead leading to drops of 9%, 17%, 8% and 1%, 3%, 2% in terms of ROUGE 1,2,L on the S2S and Unified models, respectively. As for the baseline model `Proto`, we found that it directly **copies** from the prototype summary as generated summary, which leads to a totally useless and incorrect summary.

For the human evaluation, we asked annotators to rate each summary according to its consistency and fluency. The rating score ranges from 1 to 3, with 3 being the best. Table 4 lists the average scores of each model, showing that PESG outperforms the other baseline models in both fluency and consistency. The kappa statistics are 0.33 and 0.29 for fluency and consistency respectively, and that indicates the moderate agreement between annotators. To prove the significance of these results, we also conduct the paired student t-test between our model and `Re³Sum` (row with shaded background). We obtain a p-value of 2×10^{-7} and 9×10^{-12} for fluency and consistency, respectively.

We also analyze the effectiveness of performance by the two hyper-parameters: η and ϵ. It turns out that our model has a consistently good performance, with ROUGE-1, ROUGE-2, ROUGE-L scores above 39.5, 27.5, 39.4, which demonstrates that our model is very robust.

6.2 Ablation Study

The ROUGE scores of different ablation models are shown in Table 5. All ablation models perform worse than PESG in terms of all metrics, which demonstrates the preeminence of PESG. More importantly, by this controlled experiment, we can verify the contributions of each modules in PESG.

6.3 Analysis of Editing Generator

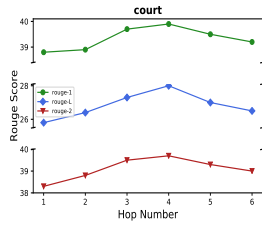We visualize the editing gate (illustrated in Equation 16) of two **randomly sampled** examples,

court

	R1	R2	RL
PESG-FC	38.7	26.2	38.6
PESG-PR	37.3	24.6	37.0
PESG-SS	38.6	25.6	38.3
PESG-FG	38.8	25.9	38.5

Figure 5: Visualizations of ROUGE score with different hop numbers.

Table 5: ROUGE scores of different ablation models of PESG.

shown in Figure 4. A lower weight (lighter color) means that the word is more likely to be copied from the summary pattern; that is to say, this word is a universal patternized word. We can see that the phrase 本院认为 (the court held that) has a lower weight than the name of the defendant (PERS), which is consistent with the fact that (the court held that) is a patternized word and the name of the defendant is closely related to the input document.

We also show a case study in Table 6, which includes the input document and reference summary with the generated summaries. Underlined text denotes a grammar error and a strike-through line denotes a fact contrary to the input document. We only show part of the document and summary due to limited space; the full version is shown in Appendix. As can be seen, the summary generated by `Uni-model` faces an inconsistency problem and the summary generated by Re^3Sum is contrary to the facts described in the input document. However, PESG overcomes both of these problems and generates an accurate summary with good grammar and logic.

6.4 Analysis of Fact Extraction Module

We investigate the influence of the iteration number when facts are extracted. Figure 5 illustrates the relationship between iteration number and the f-value of the ROUGE score. The results show that the ROUGE scores first increases with the number of hops. After reaching an upper limit it then begins to drop. This phenomenon demonstrates that the fact extraction module is effective by polishing the facts representation.

7 Conclusion

In this paper, we propose a framework named *Prototype Editing based Summary Generator* (PESG), which aims to generate summaries in formal

安徽省人民法院作出刑事判决，以斗殴罪判处被告人PERS 有期徒刑YEARS 。合肥市监狱提出减刑意见。执行机关以PERS 在服刑期间确有悔改表现，已获得奖励二个，建议对其减刑。经审理查明，罪犯PERS 在服刑期间，能认罪服法，努力参加技术学习。(Anhui Provincial People's Court made a criminal judgment and sentenced the defendant PERS to YEARS imprisonment for the crime of fighting. Hefei Prison proposed a commutation sentence since PERS had repentance and received two awards during his sentence. It was found through trial that the criminal PERS pleaded guilty and strived to participate in technical learning while serving his sentence.)

reference 本院认为，罪犯PERS 在服刑期间确有悔改，表现符合法定减刑条件。依据《中华人民共和国刑法》第NUM 款之规定，对罪犯PERS 减去自本裁定送达之日起至DATE 止尚未执行完毕的刑罚。本裁定送达后即发生法律效力。(The court believed that the criminal PERS did have repentance during his sentence, and his performance was in line with the statutory commutation conditions. According to the provisions of Section NUM of the Criminal Law of the People's Republic of China, the court decided to exempt PERS from penalty that had not been executed since the date of delivery of this ruling until DATE. Legal effect would occur upon the delivery of this ruling.)

Uni-model 本院认为，被告人PERS 以非法占有为目的，秘密窃取他人财物，数额较大，其行为已构成盗窃罪。公诉机关指控被告人PERS 盗窃罪的事实清楚，证据确实充分，本院予以支持。(The court held that the defendant PERS ~~secretly stole the property of others for the purpose of illegal possession, the stolen amount was relatively large, and his behavior had constituted theft.~~ The fact that defendant PERS committed the crime of fighting was clear, the evidence was indeed sufficient, and the court supported it.)

Re³Sum 本院认为，PERS 在道路上驾驶机动车,其行为已构成上诉人服刑确系罪。公诉机关指控的犯罪事实清楚，证据确实充分。(The court held that PERS was ~~driving a motor vehicle on the road~~, and his behavior constituted that the appellant was guilty of serving a sentence. The criminal facts accused by the public prosecution agency were clear and the evidence was indeed sufficient.)

PESG 本院认为，罪犯PERS 在服刑期间，确有悔改表现。符合法定减刑条件，依照《中华人民共和国刑法》第NUM 条之规定，裁定如下对罪犯PERS 减去有期徒刑MONTHS 。本裁定送达后即发生法律效力。(The court held that the criminal PERS did have repentance during his sentence. In accordance with the provisions of NUM and NUM of the Criminal Law of the People's Republic of China, the ruling was as follows: exempted PERS from the MONTHS penalty. Legal effect would occur upon the delivery of this ruling.)

Table 6: Examples of the generated natural answers by PESG and other models.

writing scenarios, where summaries should conform to a patternized style. Given a prototype document-summary pair, our model first calculates the cross dependency between the prototype document-summary pair. Next, a fact extraction module is employed to extract facts from the document, which are then polished. Finally, we design an editing-based generator to produce a summary by incorporating the polished fact and summary pattern. To ensure that the generated summary is consistent with the input document, we propose a fact checker to estimate the mutual information between the input document and generated summary. Our model outperforms state-of-the-art methods in terms of ROUGE scores and human evaluations by a large margin, which demonstrates the effectiveness of PESG.

like to thank Anna Hennig in Inception Institute of Artificial Intelligence for her help on this paper.

A SRU Cell

Gated recurrent unit (GRU) (Cho et al., 2014) is a gating mechanism in recurrent neural networks, which incorporate an update gate in an RNN. We first give the details of the original GRU here.

$$u_i = \sigma(W^{(u)}x_i + U^{(u)}h_{i-1} + b^{(u)}), \quad (29)$$

$$r_i = \sigma(W^{(r)}x_i + U^{(r)}h_{i-1} + b^{(r)}), \quad (30)$$

$$\tilde{h}_i = \tanh(W^{(h)}x_i + r_i \circ Uh_{i-1} + b^{(h)}), \quad (31)$$

$$h_i = u_i \circ \tilde{h}_i + (1 - u_i) \circ h_{i-1}, \quad (32)$$

where σ is the sigmoid activation function, $W^{(u)}, W^{(r)}, W^{(h)} \in \mathbb{R}^{n_H \times n_I}, U^{(u)}, U^{(r)}, U \in \mathbb{R}^{n_H \times n_H}$, n_H is the hidden size, and n_I is the size of input x_i. In the original version of GRU, the update gate u_i in Equation 29 is used to decide how much of the hidden state should be retained and how much should be updated. In our case, we want to decide which facts are salient according to the polished facts q_{k-1} at the k-th hop. To achieve this, we replace the calculation of u_i with the newly computed update gate g_i:

$$f_i = [x_i \circ q_{k-1}; x_i; q_{k-1}], \quad (33)$$

$$z_i = W^{(2)} \tanh(W^{(1)}f_i + b^{(1)}) + b^{(2)}, \quad (34)$$

$$g_i = \frac{\exp(z_i)}{\sum_{j=1}^{n_s} \exp(z_j)}, \quad (35)$$

where $W^{(2)}, W^{(1)}, b^{(1)}, b^{(2)}$ are all trainable parameters and k is the hop number in the multi-hop situation which is a hyper-parameter manually set. The effectiveness of this hyper-parameter is verified in the experimental results shown in § 6.4. Equation 32 now becomes:

$$h_i = g_i \circ \tilde{h}_i + (1 - g_i) \circ h_{i-1}. \quad (36)$$

We use the name "SRU" to denote this modified version of an GRU cell.

B Case Study

	安徽省合肥市包河区人民法院作出刑事判决，以斗殴罪判处被告人PERS 有期徒刑YEARS 。案经本院二审，准许上诉人撤回上诉。判决生效后交付执行。执行机关合肥市义城监狱于DATE 提出减刑意见，报送本院审理。本院依法组成合议庭进行了审理，现已审理终结。执行机关以罪犯PERS 在服刑期间确有悔改表现，已获得奖励二个，建议对其减刑。经审理查明，罪犯PERS 在服刑期间，能认罪服法，努力参加技术学习，积极完成生产任务。(The People's Court of Baohe District, Hefei City, Anhui Province made a criminal judgment and sentenced the defendant PERS to YEARS imprisonment for the crime of fighting. In the second instance of the court, the appellant was allowed to withdraw the appeal. The judgment takes effect after being delivered. The enforcement agency Hefei City Yicheng Prison put forward a commutation opinion on DATE and submitted it to the court for trial. The court had formed a collegiate bench in accordance with the law and had now end the trial. Criminal PERS had repentance during his sentence and received two awards, thus the enforcement agency suggested he should be commuted. Investigation confirmed that the criminal PERS was able to plead guilty while serving his sentence, strived to participate in technical studies, and completed production tasks.)
reference	本院认为，罪犯PERS 在服刑期间确有悔改表现符合法定减刑条件。依据《中华人民共和国刑法》第NUM 款之规定，裁定如下对罪犯PERS 减去自本裁定送达之日起至DATE 止尚未执行完毕的刑罚。本裁定送达后即发生法律效力。(The court held that the criminal PERS did have repentance during the period of serving his sentence in accordance with the statutory commutation conditions. According to the provisions of NUM Section of the Criminal Law of the People's Republic of China, the court decided to exempt PERS from penalty that had not been executed since the date of delivery of this ruling until DATE. Legal effect would occur upon the delivery of this ruling.)
Unified	本院认为，被告人PERS 以非法占有为目的，秘密窃取他人财物，数额较大，其行为已构成盗窃罪，依法应予惩处。公诉机关指控被告人PERS 聚众斗殴罪的事实清楚，证据确实充分，指控罪名成立，本院予以支持。被告人PERS 曾因故意犯罪被判处有期徒刑，在刑罚执行完毕后YEARS 内再犯，应当判处有期徒刑以上刑罚之罪，是累犯，依法应当从重处罚。(The court held that the defendant PERS secretly stole the property of others for the purpose of illegal possession, and the amount was relatively large. His behavior had constituted theft and should be punished according to law. The fact that the defendant PERS commiting the crimes of the fighting was clear, the evidence was indeed sufficient, thus the charges were established and the court supported it. The defendant PERS was sentenced to fixed-term imprisonment for intentional crimes. If he committed another crime in YEARS after the execution of the penalty, he should be sentenced to a fixed-term imprisonment or more. He is a recidivist and should be severely punished according to law.)
Re³Sum	本院认为NUM PERS 在道路上驾驶机动车其行为已构成上诉人服刑体系罪公诉机关指控的犯罪事实清楚证据确实充分指控的罪名成立本院予以支持被告人PERS 归案后如实供述自己的罪行依法予以从轻处罚公诉机关的量刑建议适当本院予以采纳此依照《中华人民共和国刑法》第NUM 款之规定判决如下被告人PERS 犯服刑判处有期徒刑MONTHS并处罚金人民币MONEY (The court held that NUM PERS was driving a motor vehicle on the road, and his behavior constituted that the appellant was guilty of serving a sentence. The criminal facts accused by the public prosecution agency were clear, the evidence was indeed sufficient, and the charges were supported by the court. After confessing his crimes, he should be given a lighter punishment according to law. The public prosecution agency's sentencing recommendations were appropriate and adopted by the court. According to the provisions of Section NUM of the Criminal Law of the People's Republic of China, the judgment was as follows, the defendant PERS was sentenced to imprisonment and sentenced to fixed-term MONTHS imprisonment, and fined the penalty RMB MONEY.)
PESG	本院认为，罪犯PERS 在服刑期间，确有悔改表现。符合法定减刑条件，依照《中华人民共和国刑法》第NUM 条、第NUM 条之规定，裁定如下：对罪犯PERS 减去有期徒刑MONTHS。本裁定送达后即发生法律效力。(The court believed that the criminal PERS did have repentance during his sentence. In accordance with the statutory commutation conditions and NUM and NUM of the Criminal Law of the People's Republic of China, the ruling was as follows: exempted PERS from the MONTHS penalty. Legal effect would occur upon the delivery of this ruling.)

Table 7: Examples of the generated natural answers by PESG and other models.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Deng Cai, Yan Wang, Victoria Bi, Zhaopeng Tu, Xiaojiang Liu, Wai Lam, and Shuming Shi. 2018. Skeleton-to-response: Dialogue generation guided by retrieval memory. *arXiv preprint arXiv:1809.05296*.

Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2018. Retrieve, rerank and rewrite: Soft template based neural summarization. In *ACL*, pages 152–161. Association for Computational Linguistics.

Xiuying Chen, Zhangming Chan, Shen Gao, Meng-Hsuan Yu, Dongyan Zhao, and Rui Yan. 2019. Learning towards abstractive timeline summarization. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4939–4945. International Joint Conferences on Artificial Intelligence Organization.

Xiuying Chen, Shen Gao, Chongyang Tao, Yan Song, Dongyan Zhao, and Rui Yan. 2018. Iterative document representation learning towards summarization with polishing. In *EMNLP*, pages 4088–4097.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*.

John C. Duchi, Elad Hazan, and Yoram Singer. 2010. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159.

Shen Gao, Xiuying Chen, Piji Li, Zhaochun Ren, Lidong Bing, Dongyan Zhao, and Rui Yan. 2019a. Abstractive text summarization by incorporating reader comments. In *AAAI*.

Shen Gao, Zhaochun Ren, Yihong Eric Zhao, Dongyan Zhao, Dawei Yin, and Rui Yan. 2019b. Product-aware answer generation in e-commerce question-answering. In *WSDM*.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *NAACL-HLT*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *CoRR*, abs/1603.06393.

Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *TACL*, 6:437–450.

R. Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Adam Trischler, and Yoshua Bengio. 2018. Learning deep representations by mutual information estimation and maximization. *CoRR*, abs/1808.06670.

Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. 2018. Learning to write with cooperative discriminators. In *ACL*.

Wan-Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. In *ACL*.

Wenpeng Hu, Zhangming Chan, Bing Liu, Dongyan Zhao, Jinwen Ma, and Rui Yan. 2019. Gsn: A graph-structured network for multi-party dialogues. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5010–5016. International Joint Conferences on Artificial Intelligence Organization.

Byeongchang Kim, Hyunwoo Kim, and Gunhee Kim. 2019. Abstractive Summarization of Reddit Posts with Multi-level Memory Networks. In *NAACL-HLT*.

Piji Li, Wai Lam, Lidong Bing, and Zihao Wang. 2017. Deep recurrent generative decoder for abstractive text summarization. In *EMNLP*, pages 2091–2100.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Shuming Ma, Xu Sun, Junyang Lin, and Houfeng Wang. 2018. Autoencoder as assistant supervisor: Improving text representation for chinese social media text summarization. In *ACL*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018a. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *EMNLP*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018b. Ranking sentences for extractive summarization with reinforcement learning. In *NAACL-HLT*.

Tatsuro Oya, Yashar Mehdad, Giuseppe Carenini, and Raymond Ng. 2014. A template-based abstractive meeting summarization: Leveraging summary and source text relationships. In *INLG*, pages 45–53.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *ICML*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *ICLR*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *EMNLP*, pages 379–389.

Natalie Schluter. 2017. The limits of automatic summarisation according to rouge. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 41–45. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *ACL*.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Chongyang Tao, Shen Gao, Mingyue Shang, Wei Chung Wu, Dongyan Zhao, and Rui Yan. 2018. Get the point of my utterance! learning towards effective responses with multi-head attention mechanism. In *IJCAI*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *NIPS*, pages 2692–2700.

Yu Ping Wu, Furu Wei, Shaohan Huang, Zhoujun Li, and Ming Zhou. 2018. Response generation by context-aware prototype editing. *CoRR*, abs/1806.07042.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2018. Sequential copying networks. In *AAAI*.