Summarizing Procedural Text: Data and Approach

Shen Gao [♠][*], **Haotong Zhang** [△][*], **Xiuying Chen** [♡], **Dongyan Zhao** [◇][†] **and Rui Yan** [♣][†]
[♠]School of Computer Science and Technology, Shandong University
[◇]Wangxuan Institute of Computer Technology, Peking University
[♣]Gaoling School of Artificial Intelligence, Renmin Univ. of China
[△]Department of Mathematics, National University of Singapore
[♡]King Abdullah University of Science and Technology
shengao@sdu.edu.cn, haotongz@u.nus.edu, xiuying.chen@kaust.edu.sa,
zhaody@pku.edu.cn, ruiyan@ruc.edu.cn

Abstract

Procedural text is a widely used genre that contains many steps of instructions of how to cook a dish or how to conduct a chemical experiment and analyzing the procedural text has become a popular task in the NLP field. Since the procedural text can be very long and contains many details, summarizing the whole procedural text or giving an overview for each complicated procedure step can save time for readers and help the reader to capture the core action in the procedure. In this paper, we propose the procedural text summarization task with two summarization granularity: step-view and global-view, which summarizes each step in procedural text separately or gives an overall summary for all steps respectively. To tackle this task, we propose an **E**ntity-**S**tate **G**raph-based **S**ummarizer (ESGS) which is based on state-of-the-art entity state tracking methods and constructs a heterogeneous graph to aggregate contextual information for each procedure. In order to help the summarization model focus on the salient entity, we propose to use the contextualized procedure graph representation to predict the salient entity. Experiments conducted on two datasets verify the effectiveness of our proposed model, and the code and datasets will be released on `https://github.com/gsh199449/procedural-summ`.

1 Introduction

Procedural texts, *e.g.,* scientific articles, instruction books, or recipes, are widely spread and useful in many real-world applications (Tang et al., 2020; Gupta and Durrett, 2019a; Du et al., 2019b). In procedural text modeling field, many research works focus on entity state tracking (Gupta and Durrett, 2019a,b; Dalvi et al., 2018; Swarup et al., 2020) and reasoning (Tandon et al., 2018), and how to summarize the procedural text has not been fully

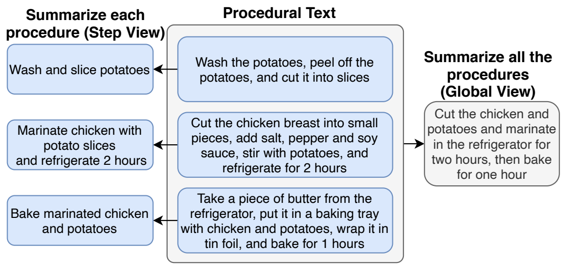
Summarize each procedure (Step View)
Procedural Text
Summarize all the procedures (Global View)
Wash and slice potatoes
Wash the potatoes, peel off the potatoes, and cut it into slices
Marinate chicken with potato slices and refrigerate 2 hours
Cut the chicken breast into small pieces, add salt, pepper and soy sauce, stir with potatoes, and refrigerate for 2 hours
Bake marinated chicken and potatoes
Take a piece of butter from the refrigerator, put it in a baking tray with chicken and potatoes, wrap it in tin foil, and bake for 1 hours
Cut the chicken and potatoes and marinate in the refrigerator for two hours, then bake for one hour

Figure 1: Example of procedural text summarization with two sub-tasks for different granularity summary.

explored. Since the procedural text contains many steps and the procedure is usually long, summarizing the procedural text can save time for readers when they want to quickly locate the useful step or take an overview of the procedural text. In this paper, we propose a new summarization task: *Procedural Text Summarization*. Intuitively, there are two sub-tasks (shown in Figure 1): (1) summarize each procedure (**Step-View**); (2) summarize all the procedures into a comprehensive summary (**Global-View**). The first one aims to summarize the main action in a procedure by incorporating contextual information from related procedures, and the second one aims to capture the salient steps from all procedures by leveraging the structure (*a.k.a.,* relationship) of all procedures.

In each procedure, the main content is the description of actions, *e.g.,* cut the whole potato into slices, heat the iron at room temperature to 1500 degrees. And these actions usually cause the state changes of the entity, *e.g.,* from room temperature state change to 1500 degrees of the entity iron. Thus, the core of the procedural text summarization model is to capture the salient entity and describe the trace of entity state changes. In order to generate a better summary for procedural text, there are two challenges which should be tackled: (1) the first challenge is modeling the relationship between procedures to explore entity states in past and future and capture the comprehensive contex-

[*] Equal contribution.
[†] Corresponding author.

tual information; and (2) the second is to identify the salient entity for the procedure. However, the plain text summarization methods (Zhang et al., 2020b; Zhu et al., 2021) only incorporates specific procedure text and cannot model the relationship between contextual procedures. Many research works (Du et al., 2019a; Tandon et al., 2018) on procedural text focus on extracting the state changes of each entity involved in the process. Based on these existing entity state change analysis methods, to capture the salient procedure and tackle the procedural text summarization task, we propose to employ the trace of entity state changes to explicitly construct the relationship between procedures.

In this paper, we propose a procedural text summarization framework named **E**ntity-**S**tate **G**raph-based **S**ummarizer (ESGS) that constructs a heterogeneous graph with *procedure nodes* and *entity nodes*. To construct the relationship between nodes, we follow the existing procedural text state tracking method (Tandon et al., 2020) which is based on GPT-2 (Radford et al., 2019). Then we propose the entity state-aware message passing method on the graph to understand the procedural text in a comprehensive perspective. In order to identify the salient entity, we propose an entity selection module by using the graph representation of procedure. Finally, we employ a pre-trained language model to incorporate the graph and salient entity representation to generate the summary for procedural text. To verify the effectiveness of ESGS, we firstly conduct the experiment on the benchmark dataset WikiHow$_{\text{proc}}$, and we also propose a new procedural text summarization dataset PsyStory which summarizes the procedural text in global-view. Extensive experiments on these two datasets demonstrate that the ESGS brings substantial improvements over several strong baselines including state-of-the-art summarization method.

To sum up, our contributions can be summarized as follows:

- We propose a procedural text summarization task that aims to generate two granularity of summaries for each procedure and all procedural text.
- We propose to leverage the entity state tracking method to construct a heterogeneous graph, and then generate a summary by incorporating the salient entity and graph representation.
- We propose a new procedural text summarization dataset PsyStory.
- Experiments conducted on two datasets show that our ESGS method outperforms all baselines, including the state-of-the-art summarization model.

2 Related Work

2.1 State Tracking in Procedural Text

Procedural text is a domain of text involved with understanding some kind of process, such as a phenomenon arising in nature or a set of instructions to perform a task. Entity tracking is the core of understanding the procedural text. The goal is to track the sequence of state changes (*e.g.,* creation and movement) entities undergo over long sequences of procedure steps. Dalvi et al. (2019) propose to use the WikiHow data to train the state change tracking model with limited states, which is an open-domain procedural text dataset. Past work involves both modeling entities across procedure steps (Das et al., 2019; Tang et al., 2020; Kiddon et al., 2015; Gupta and Durrett, 2019b). Tandon et al. (2020) firstly propose a GPT-2 based entity state tracking method which can be used to analyze the open-domain procedural text with unlimited state space.

2.2 Text Summarization

Abstractive summarization methods (Gehrmann et al., 2018a; Jin and Wan, 2020; Maynez et al., 2020; Liu and Liu, 2021) aims to generate a fluent and condensed short text to cover the main idea of the input document. Many researchers use the sequence-to-sequence based framework to read the document first and generate a summary by decoder (See et al., 2017; Lin et al., 2018; Celikyilmaz et al., 2018). With the development of pretraining techniques, the fluency of abstractive summarization has been significantly improved (Lewis et al., 2020; Zhang et al., 2020b) by using large-scale plain text. However, most of the existing summarization research works concentrate on summarizing plain documents (Gao et al., 2020), and the genre of procedural which is usually long and contains many detailed facts has not been fully explored in the summarization research field. Although Koupaee and Wang (2018) propose to use the WikiHow as the summarization dataset, the research works on this dataset concatenate all the steps in procedural text and treat it as the plain document summarization task.

3 Problem Formulation

Given a procedural text $P = \{s_1, \cdots, s_{L_p}\}$ with L_p procedures and each procedure $s_i =$

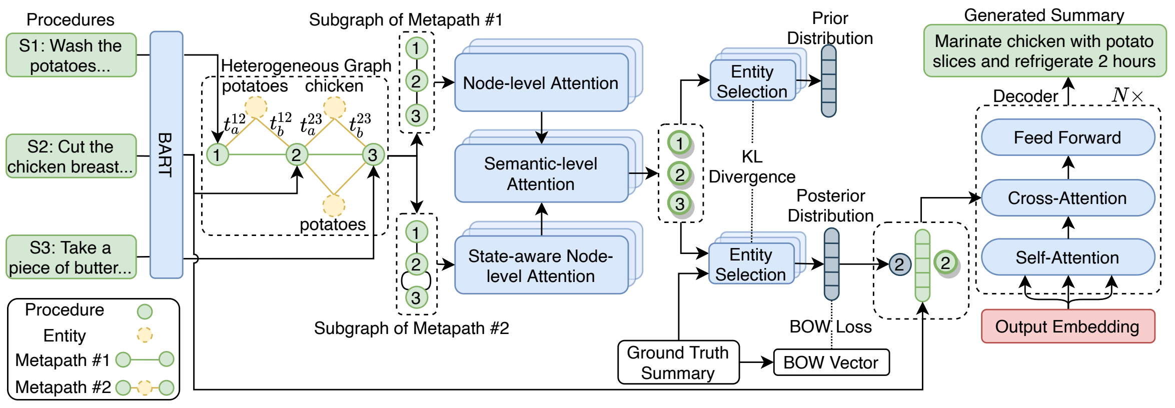

Figure 2: Overview of ESGS with four parts: (1) Procedural Graph Construction (refer to Figure 3 for more details); (2) Procedural Graph Encoding; (3) Entity Selection Module; and (4) Summary Generation. And we use the step-view summarization process for 2-nd procedure as an example.

$\{w_{i,1}, \cdots, w_{i,L_s}\}$ contains L_s words. In step-view sub-task, our goal is to generate the summary for each procedure $\hat{Y}^i = \{\hat{y}^i_1, \cdots, \hat{y}^i_{L_y}\}$, where $\hat{Y}^i$ is for procedure s_i and it has L_y is the number of words of the summary. And in the global-view, we aim to generate a summary for all the procedures in P. Finally, we use the difference between generated summary and the ground truth as the training objective. In the following sections, we use the step-view as the example to illustrate our method, and the model difference for global-view is introduced in § 4.7.

4 ESGS Model

4.1 Overview

In this section, we introduce the **E**ntity-**S**tate **G**raph-based **S**ummarizer (ESGS). Figure 2 shows an overview of ESGS which has four main parts:

• **Procedural Graph Construction** uses the GPT-2 based method to analyze the entity state changes in the procedure and uses the trace of entity state to construct the graph.

• **Procedural Graph Encoding** employs the state-aware message passing method to model the contextual information for each procedure.

• **Entity Selection Module** uses the posterior information to predict salient entity in the procedure.

• **Summary Generation** employs the pre-trained BART and uses a graph attention layer to incorporate the selected entity and the contextual information of procedure in the summary generation.

Since there are two procedural text summarization sub-tasks: step-view and global-view summarization. In the following sections, we use the step-view procedural text summarization task as the example to illustrate the details of the ESGS model. Then the step-view ESGS model can be easily adapt to the global-view summarization task with only two small modifications, and we will illustrate this variant model in § 4.7.

4.2 Preliminary

Heterogeneous Graph. In ESGS, we employ a heterogeneous graph to model the relationship between procedures and entities, which is an information network with two types of nodes and edges (Sun and Han, 2013; Wang et al., 2019). A heterogeneous graph, denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, consists of a node set $\mathcal{V}$ and an edge set $\mathcal{E}$, and associates with a node type mapping function $\phi : \mathcal{V} \to \mathcal{A}$ and an edge type mapping function $\psi : \mathcal{E} \to \mathcal{R}$, where $\mathcal{A}$ and $\mathcal{R}$ denote the node and edge types. In Figure 3, we construct a heterogeneous graph with two types of node: procedure and entity, and three types of links.

Metapath (Sun et al., 2011). A metapath ρ is defined as a path in the form of $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \cdots \xrightarrow{R_{l-1}} A_l$, which describes a composite relation between node type A_1 and A_l. Each metapath may have multiple **metapath instances**. As shown in Figure 2, two procedures can be connected via two *metapath*s, *e.g.,* "procedure-procedure", and "procedure-entity-procedure", where metapath "procedure-entity-procedure" has 3 *metapath instance*s which is shown using the yellow line.

4.3 Procedural Graph Construction

We first leverage an open-domain entity state tracking method *ProcGPT* (Tandon et al., 2020) to analyze the procedural text, which is based on the

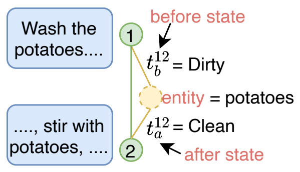

Figure 3: Graph construction from procedural text. Graph contains two types of nodes: sentence and entity. The states of entity store in the edge.

pre-training language model GPT-2 (Radford et al., 2019). Thus, we employ this method to extract the entity and states from a procedure step text as a set of tuples "(entity e^i, before-state t_b^i, after-state t_a^i)":

$$(e^i, t_b^i, t_a^i) = \text{ProcGPT}(s_i), \tag{1}$$

where s_i is the i-th step procedure text. We finetune the ProcGPT (Tandon et al., 2020) using the procedure text and entity tuple parallel training data to generate the tuples for a procedure step text s_i. Figure 3 shows an example of graph construction, we first use the finetuned ProcGPT to extract entity-states tuples. Then we use the procedure step text and entity word as two types of graph node, and build the edge between entity and its states with the procedure step node. For brevity, we only use one entity and state for a procedure step to illustrate the model in following sections, and there can be more than one entity for each step in the real datasets.

After obtaining the entity and the corresponding state in each procedure, we employ these relationships to build a graph for the procedural text. Intuitively, we have three types of edges in the graph: (1) edge between two adjacent procedures; (2) edge from procedure node to entity node; (3) edge from entity node to procedure node. And we store the "after state" on the (2) edge and store the "before state" on the (3) edge.

4.4 Procedural Graph Encoding

First, we employ the pre-trained BART (Lewis et al., 2020) encoder to transform the procedure text into vector representations:

$$\begin{aligned}\{\mathbf{h}_{i,0}, \mathbf{h}_{i,1}, \cdots, \mathbf{h}_{i,L_s}\} = \\ \text{Enc}([\text{CLS}], w_{i,1}, \cdots, w_{i,L_s}),\end{aligned} \tag{2}$$

where Enc is the BART encoder which outputs the vector representation $\mathbf{h}_{i,j}$ of j-th input word $w_{i,j}$ in i-th procedure. To obtain a vector representation of each procedure, we extract the hidden state $\mathbf{h}_{i,0}$ of the special token [CLS] as the representation $\mathbf{s_i} = \mathbf{h}_{i,0}$ of i-th procedure. We use the $\mathbf{P} = \{\mathbf{s_1}, \dots, \mathbf{s_{L_p}}\}$ to denote the representations for all procedures. Similarly, we use the same method to encode the entity word and its states into vector representations:

$$\mathbf{e^i} = \text{Enc}(e^i), \mathbf{t_b^{ij}} = \text{Enc}(t_b^{ij}), \mathbf{t_a^{ij}} = \text{Enc}(t_a^{ij}). \tag{3}$$

where e^i is an entity word in i-th procedure, and t_a^{ij}, t_b^{ij} denote the "after state" and "before state" between i-th and j-th procedure.

To capture the contextual information for a procedure step, inspired by the Heterogeneous graph Attention Network (HAN) (Wang et al., 2019), we propose an entity state-aware graph encoding method that integrates the state into the message passing between the nodes. Before conducting message passing between nodes in the multi-layer graph, we use the $\mathbf{s_i}, \mathbf{e^i}, \mathbf{t_b^{ij}}, \mathbf{t_a^{ij}}$ as the initial node representation for procedure, entity and state, respectively. Similar to the HAN, our method also follows a hierarchical attention structure: from node-level to semantic-level. Node-level attention assigns different weights for neighbor nodes on metapath. We extract two homogeneous sub-graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ which are constructed from the original heterogeneous graph $\mathcal{G}$. $\mathcal{G}_1$ and $\mathcal{G}_2$ are the sub-graphs in which procedure nodes are connected by metapath "procedure-procedure" and "procedure-entity-procedure", respectively. Specifically, in original heterogeneous graph $\mathcal{G}$, an edge $s_i \xrightarrow{t_a^{ij}} e^{ij} \xrightarrow{t_b^{ij}} s_j$ becomes a new edge $s_i \xrightarrow{\mathbf{t_a^{ij}}||\mathbf{t_b^{ij}}} s_j$ in $\mathcal{G}_2$ with attribute $[\mathbf{t_a^{ij}}||\mathbf{t_b^{ij}}]$ where $||$ denotes the concatenate operation.

First, the node representation $\mathbf{s_i}$ is mapped to a feature space by transformation matrix $M_l, l = \{1, 2\}$ for each sub-graph:

$$\mathbf{s_i}'^l = M_l \cdot \mathbf{s_i}. \tag{4}$$

Then the model learns the weight a_{ij}^l for each neighbour node pair s_i and s_j in sub-graph $\mathcal{G}_l$. In $\mathcal{G}_1$, the procedure nodes are connected sequentially, the weight is calculated by the node representations:

$$a_{ij}^1 = \text{Attn}(\mathbf{s_i}'^l, \mathbf{s_j}'^l), \tag{5}$$

where Attn denotes the multi-head (Vaswani et al., 2017) node-level attention operation. In $\mathcal{G}_2$, since the procedure nodes are connected by the shared entity and the state trace of entity is stored on the edge, we incorporate the edge attributes information to calculate the weights between nodes:

$$a^2_{ij} = \text{MLP}([\mathbf{t^{ij}_a} || \mathbf{t^{ij}_b}]), \tag{6}$$

where MLP is a fully connected layer with activation function.

After obtaining a^l_{ij} between procedure i and j, we normalize them to get attention weight α^l_{ij}:

$$\alpha^l_{ij} = \text{softmax}(a^l_{ij}). \tag{7}$$

Then the updated representation $\mathbf{z_i}^l$ of node s_i is aggregated by the weighted sum of all neighbour nodes in sub-graph $\mathcal{G}_l$:

$$\mathbf{z_i}^l = \sigma(\textstyle\sum_{j \in \mathcal{N}^l_i} \alpha^l_{ij} \cdot \mathbf{s_j}'^l), \tag{8}$$

where σ is an activation function, $\mathcal{N}^l_i$ represents neighbour nodes of i in sub-graph $\mathcal{G}_l$. Next, we combine the procedure node representations $\{\mathbf{z_i}^1, \mathbf{z_i}^2\}$ for different metapaths into an overall representation $\mathbf{z_i}$ by using semantic-level attention. We use the same semantic-level attention as the original HAN, and we refer readers to the HAN (Wang et al., 2019) for more details.

4.5 Entity Selection Module

To generate a concise summary for the procedure which captures the main actions, we should select the salient entity from the procedure text. When the model focuses on different entities, a different summary can be generated. In this paper, we propose to use a entity selection module to predict salient entity to help the summarization model focus on main procedure actions.

When training the summarization model, if we predict the entity selection only based on the procedure (*a.k.a.,* prior information) without knowing the ground truth summary (*a.k.a.,* posterior information), it is difficult to generate a better summary since a salient entity might not be selected accurately. It will be sub-optimal to train the summarization model by selecting trivial entities since it cannot provide any helpful training signals (Lian et al., 2019). In contrast, if we use procedure text and the ground truth summary to predict the posterior distribution over entities, it can provide an effective training signal since the ground truth summary contains the salient entities.

In this paper, we propose to select the salient entity by using both prior and posterior information. We first encode all the entities into vectors $e = \{e_1, \dots, e_{L_e}\}$ using Equation 2, where e is the entity set for i-th procedure with L_e entities, and we omit the subscript i for brevity. In the prior entity distribution, we define a conditional probability distribution over all the entities e using the procedure text s_i, denoted by $p(e|s_i)$. Specifically, we incorporate two types of information to model the prior entity distribution $p(e|s_i)$: (1) the graph representation $\mathbf{z_i}$ for procedure s_i with the contextual information of the related procedures; (2) the vector representation $\mathbf{s_i}$ which is encoded by the pre-trained language model.

$$p(e|s_i) = \text{MHAttn}(\{e_1, \dots, e_{L_e}\}, [\mathbf{s_i} \oplus \mathbf{z_i}]), \tag{9}$$

where $\oplus$ denotes the vector concatenation, and MHAttn denotes the multi-head attention mechanism (Vaswani et al., 2017) to measure the relationship between each entity and the procedure s_i Then, we use the prior entity distribution to weighted sum the entity representations as selected entity representation $\mathcal{E}^{\text{prior}}$:

$$\mathcal{E}^{\text{prior}} = \textstyle\sum_{j=1}^{L_e} e_j p(e_j|s_i). \tag{10}$$

In the posterior entity distribution, we calculate it by adding the ground truth summary Y^i, denoted by $p(e|s_i, Y^i)$. We use the same BART encoder (Equation 2) to obtain the representation $\mathbf{Y}^i$ of the ground truth summary Y^i.

$$\begin{aligned} &p(e|s_i, Y^i) = \\ &\text{MHAttn}(\{e_1, \dots, e_{L_e}\}, [\mathbf{s_i} \oplus \mathbf{z_i} \oplus \mathbf{Y}^i]). \end{aligned} \tag{11}$$

Similarly, we also obtain the selected entity representation $\mathcal{E}^{\text{post}}_i$ by posterior distribution using the same method as Equation 10. Different from the prior information, the posterior information contains more accurate entity selection information which is predicted by the ground truth summary.

In the training phase, we employ the selected entity representation by the posterior distribution in the summary generation, and in the testing, we use the selected entity representation by the prior distribution since the ground truth summary is not available. Intuitively, there is a discrepancy between prior and posterior that will lead to the mismatch between the training and testing. Thus, we employ the KL divergence as the training objective to minimize the distance between between the prior and posterior distribution:

$$\mathcal{L}^i_{\text{KL}} = \sum_{j=1}^{L_e} p\left(\mathbf{e}_j | s_i, Y^i\right) \log \frac{p\left(\mathbf{e}_j | s_i, Y^i\right)}{p\left(\mathbf{e}_j | s_i\right)}. \tag{12}$$

Inspired by Zhao et al. (2017), to ensure the accuracy of the selected entity, we enforce the relevance between the selected entity and the ground truth summary. Specifically, we apply a fully connected layer which uses the selected entity representation by posterior distribution as input and predicts the the bag-of-word (BOW) of the ground truth summary Y^i:

$$p(Y^i|\mathcal{E}_i^{\text{post}}) = \text{softmax}(W\mathcal{E}_i^{\text{post}} + b), \tag{13}$$

$$\mathcal{L}_{\text{bow}}^i = \textstyle\prod_{j=1}^{L_y} p(Y^i|\mathcal{E}_i^{\text{post}}), \tag{14}$$

where W, b are the trainable parameters.

4.6 Summary Generation

We employ the pre-trained language model BART (Lewis et al., 2020) as the decoder to generate the summary. In order to incorporate the selected salient entity and the contextualized procedure representation from the graph model, we propose to insert an additional attention layer in the BART. We first apply the self-attention on the masked output summary embeddings and result in the self-attention output $\mathbf{a}^s$. This process is the same as the original Transformer (Vaswani et al., 2017) and we omit it due to the limited space. In the original BART, we should use the output $\mathbf{a}^s$ to cross-attend to the word-level procedure hidden states $\{\mathbf{h}_{i,1}, \cdots, \mathbf{h}_{i,L_s}\}$ produced by the BART-based procedure encoder (Equation 2). To aggregate salient information from both of the updated graph node $\mathbf{z_i}$ and selected entity representations $\mathcal{E}_i^{\text{post}}$ or $\mathcal{E}_i^{\text{prior}}$, we concatenate these vector representations with the word-level procedure hidden states in the cross-attention layer:

$$\mathbf{a}^q = \text{MHAttn}(\mathbf{a}^s, \{\mathbf{h}_{i,1}, \cdots, \mathbf{h}_{i,L_s}, \mathcal{E}_i^*, \mathbf{z_i}\}), \tag{15}$$

where $\mathcal{E}_i^*$ denotes the selected salient entity by prior or posterior distribution in testing or training phase respectively. Finally, we apply a fully connected feed-forward network on $\mathbf{a}^g$ to predict the distribution over the vocabulary of the generated summary. We use the cross-entropy loss $\mathcal{L}_{\text{ce}}^i$ between generated summary $\hat{Y}^i$ and ground truth summary Y^i to optimize all the parameters of ESGS, and the final loss function $\mathcal{L}^i$ for i-th procedure is defined as:

$$\mathcal{L}^i = \mathcal{L}_{\text{bow}}^i + \mathcal{L}_{\text{KL}}^i + \mathcal{L}_{\text{ce}}^i. \tag{16}$$

4.7 Model Variant for Global-View Setting

In the global-view procedural text summarization, we should summarize all the procedures instead of summarizing each procedure separately. There are two small differences in the model for global-view procedural text summarization. First, in the entity selection module, instead of selecting a salient entity for each procedure, we use all the procedure graph nodes to predict the salient entity for the whole procedural text. Second, we modify the cross attention in decoder (Equation 15) that concatenates all the procedures graph nodes representation $\{\mathbf{z_0}, \dots, \mathbf{z_{L_P}}\}$ instead of only use one procedure graph node.

	WikiHow$_{\text{proc}}$	PsyStory
Task Type	Step-View	Global-View
# of training samples	13612	4200
# of test samples	2917	900
# of validation samples	2917	900
Avg. steps	6.67	5.00
Avg. words of procedure	40.81	8.74
Avg. words of summary	9.42	14.57
Avg. entities per step	2.03	0.87
Avg. states per step	2.37	1.66

Table 1: Dataset Statistics.

5 Experimental Setup

5.1 Dataset

To validate the effectiveness of the procedural text summarization methods, we propose two datasets: WikiHow$_{\text{proc}}$ and PsyStory. Detail statistics are shown in Table 1. We show the performance of some summarization baselines on these datasets in Table 2.

WikiHow$_{\text{proc}}$ is a modified version based on WikiHow dataset (Koupaee and Wang, 2018) which contains articles describing procedural tasks about various topics (from arts and entertainment to computers and electronics) with multiple steps. Many existing procedural text state analysis methods (Tandon et al., 2020; Zhang et al., 2020c,a; Dalvi et al., 2019; Goyal et al., 2021) have conducted experiments on the WikiHow dataset, and this is the benchmark dataset on procedural text modeling. Each article consists of multiple paragraphs and each paragraph starts with a sentence summarizing it. As illustrated in previous sections, we use the ProcGPT (Tandon et al., 2020) to annotate entity and state changes of WikiHow dataset, and then remove the low-quality data samples which do not have any entity or only a few entities.

Rashkin et al. (2018) propose a story dataset with the explanation of characters' naive psychology as fully-specified chains of mental states for motivations and emotional reactions. This annotation is in sentence-level, and we can see the character as an entity and mental state as the entity state. We use an outsourcing human annotation service (we paid $1 for each data sample) with the independent annotation quality control to write the summary for procedural text. Since the service vendor have independent quality control, we deem the summary as a high-quality ground-truth summary.

5.2 Evaluation Metrics

We adopt ROUGE score (Lin, 2004) and BLEU (Papineni et al., 2002) which are widely applied for summarization and text generation evaluation (Gao et al., 2019; Chen et al., 2018). The ROUGE metrics compare generated summary with the reference summary by computing overlapping lexical units, including ROUGE-1/2 (n-gram), and ROUGE-L (longest common subsequence).

5.3 Comparison Methods

To prove the effectiveness of each module, we conduct ablation studies that remove each key module in ESGS, and then form 3 baseline methods: (1) `ESGS-MsgPass` uses the original Heterogeneous Graph Attention Network (HAN) as the graph encoder which removes the entity state-aware message passing module in ESGS. (2) `ESGS-KLLoss` removes the KL-divergence loss from the training objective and only uses the prior information in training and testing. (3) `ESGS-BOWLoss` removes the bag-of-word loss from the training objective.

Apart from the ablation study, we also compare with the following baselines: (1) `TextRank` (Mihalcea and Tarau, 2004) is a graph-based ranking model for extractive document summarization. (2) `S2SA` is the Sequence-to-Sequence framework (Sutskever et al., 2014) which is equipped with the attention mechanism (Bahdanau et al., 2015) as a baseline method. (3) `PGNet` (See et al., 2017) propose the copy mechanism to directly copy out-of-vocabulary words from input document to summary. (4) `BART` (Lewis et al., 2020) and `T5` (Raffel et al., 2020) are large-scale pre-trained language models, and we fine-tune these models on our procedural text summarization task. (5) `PEGASUS` (Zhang et al., 2020b) is a large-scale pre-trained Transformer model with a new self-supervised summarization objective, and this method achieves the state-of-the-art performance on many summarization benchmark datasets. (6)

	Method	R1	R2	RL	BLEU
WikiHow$_{proc}$	TextRank	1.10	0.25	1.04	0.03
	S2SA	7.81	0.43	7.64	0.18
	PGNet	9.47	1.08	9.06	0.32
	PEGASUS	17.08	4.71	15.98	2.71
	T5	19.92	5.61	18.98	1.28
	BottomUp	22.69	3.37	16.09	1.03
	BART	21.94	6.70	20.47	2.37
	BART+CTX	22.37	6.95	20.85	2.34
	ESGS	**23.90**	**7.56**	**22.17**	**2.73**
PsyStory	TextRank	24.28	7.00	22.35	3.34
	S2SA	11.46	0.48	9.99	0.20
	PGNet	23.29	3.68	19.83	1.68
	BottomUp	24.51	4.21	19.22	0.97
	PEGASUS	28.27	8.74	25.38	4.63
	T5	42.51	16.18	35.46	8.72
	BART	42.60	16.59	35.39	8.62
	ESGS	**44.28**	**18.31**	**36.90**	**9.95**

Table 2: Automatic metrics comparison between baselines on two datasets.

`BottomUp` (Gehrmann et al., 2018b) is an entity-driven summarization method. (7) `BART+CTX` is based on `BART` and concatenates the surrounding two procedures with the original input procedure as contextual information.

5.4 Implementation Details

The batch size is 16 with gradient accumulation to simulate a large batch size. We pad or cut the procedure to contain 200 words, and the maximum decoding length is 200. We initialize BART with BART$_{base}$[1] with 16 attention heads, 768 hidden size and 6 Transformer layers.

6 Experimental Result

6.1 Overall Performance

In Table 2, we examine the performance of our model and baseline methods on two datasets in terms of ROUGE score. We can see that ESGS achieves a 39.93%, 60.51%, and 38.74% increment over the state-of-the-art summarization method `PEGASUS` in terms of ROUGE-1, ROUGE-2, and ROUGE-L on the benchmark dataset WikiHow$_{\text{proc}}$. On the global-view procedural text summarization task, we can find that ESGS also outperforms other baseline methods in terms of three metrics. It is worth noticing that the baseline model `BART+CTX` which also incorporates the contextual information of a procedure outperforms other baselines, which demonstrates the effectiveness of using the contextual information. However, the performance

[1] `https://huggingface.co/facebook/bart-base`

	Method	R1	R2	RL	BLEU
WikiHow$_{proc}$	ESGS-MsgPass	20.81	6.23	19.93	0.9
	ESGS-KLLoss	21.34	6.61	20.18	1.72
	ESGS-BOWLoss	21.52	6.24	19.99	2.25
	ESGS	**23.9**	**7.56**	**22.17**	**2.66**
PsyStory	ESGS-MsgPass	42.55	17.11	35.71	9.61
	ESGS-KLLoss	43.22	17.84	36.28	9.50
	ESGS-BOWLoss	43.34	17.69	36.28	9.32
	ESGS	**44.28**	**18.31**	**36.90**	**9.95**

Table 3: Comparison between ablation models.

	Method	R1	R2	RL	BLEU
WikiHow$_{proc}$	ESGS-Entity	21.18	6.26	19.65	2.24
	ESGS-Proc	21.53	6.37	20.09	1.92
	ESGS	**23.90**	**7.56**	**22.17**	**2.66**
PsyStory	ESGS-Entity	42.83	17.11	36.43	9.30
	ESGS-Proc	43.39	18.30	36.86	9.50
	ESGS	**44.28**	**18.31**	**36.90**	**9.95**

Table 4: Comparison between different metapaths.

of BART+CTX is still 3.33% worse than ESGS in terms of ROUGE-1 score, which indicates that the change of entity state is important for summarizing the procedural text and such a simple concatenation method cannot fully explore the relationship between procedures. And the observation that the ESGS outperforms the BART on PsyStory also verifies this assumption, since BART also uses all the procedures in summarization.

6.2 Ablation Study

To verify the effectiveness of each module in ESGS, we conduct several ablation models (shown in § 5.3) on both datasets, and the result is shown in Table 3. All ablation models perform worse than ESGS in terms of all metrics on both datasets, which demonstrates the preeminence of ESGS. We can find that the ESGS-MsgPass performs worse among all the ablation models, and it confirms that the entity state information can help the model to identify whether the message is salient for passing among procedure nodes.

6.3 Effectiveness of Different Metapath

In our ESGS model, we employ a metapath-based heterogeneous graph that passes messages among nodes along two metapaths. In this section, we conduct experiments of removing each metapath from the model to prove the effectiveness metapaths. Table 4 shows the ROUGE scores for each ablation model. We can find that the metapath "sentence-entity-sentence" contributes most to the summarization performance since it connects disadjunct procedures using the common entity and models the transition trace of the entity state which is important for summarizing the salient entity and its state changes.

Procedures	#1. Use a soft cloth and only dampen it rather than soak it, so that the carpet or rug is not made wet, only moistened. (cloth, dry, wet) *#2. Keep to the pile of the carpet and wipe in this direction at all times. (carpet, dirty, clean), (cloth, wet, hold), (hand, empty, holding cloth)* #3. Rub warm breadcrumbs through the surface of carpet. This will bring out the colour again.
Ref.	Wipe across the carpet or rug using even, long strokes
B.+C.	Wipe the carpet
ESGS	Wipe down the surface of the carpet with a damp cloth

Table 5: Examples of the generated summary by ESGS and BART+CTX for procedure #2. Text in blue denotes the contextual information from other procedures.

6.4 Case Study

Table 5 shows a procedural text from WikiHow$_{proc}$ dataset and its corresponding summaries generated by different methods in the step-view task. Due to the limited space, we show an example with short procedures. We can observe that BART based baselines generate a fluent summary with incomplete facts. On the contrary, ESGS produces a fluent summary that is consistent with the main step of the procedural, since the relationship between step #1 and #2 is captured by the "procedure-entity-procedure" metapath.

7 Conclusion

In this paper, we propose the procedural text summarization task which aims to generate two granularity summaries (step-view and global-view). We propose to use the heterogeneous graph model **E**ntity-**S**tate **G**raph-based **S**ummarizer (ESGS) to construct the relationship between procedures using the trace of entity state changes. To focus on the salient entities, we also propose an entity selection module that is trained by using posterior information to provide effective guidance. Finally, we generate the summary by incorporating the updated graph and selected salient entities. And we conduct experiments on a modified version of benchmark dataset WikiHow$_{proc}$, and we also construct a new procedural text summarization dataset PsyStory with global-view summary. ESGS achieves the state-of-the-art performance on both datasets.

8 Limitations

Since we use a large pretrain language model as the backbone of our proposed medthod, it is hard to deploy on the edge devices or mobile phones. We can employ the model compression method to accelerate the inference speed on the server and provide services to the user through internet.

Acknowlegement

This work was supported by the National Key Research and Development Program of China (No. 2021YFC3340303), the National Natural Science Foundation of China (NSFC Grant No. 62122089 & No. 61876196), Beijing Outstanding Young Scientist Program (NO. BJJWZYJH012019100020098), and Intelligent Social Governance Platform, Major Innovation & Planning Interdisciplinary Platform for the "Double-First Class" Initiative, Renmin University of China. Rui Yan is also supported by Beijing Academy of Artificial Intelligence (BAAI).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In *NAACL*.

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

Bhavana Dalvi, Lifu Huang, Niket Tandon, Wen tau Yih, and Peter Clark. 2018. Tracking state changes in procedural text: a challenge dataset and models for process paragraph comprehension. In *NAACL*.

Bhavana Dalvi, Niket Tandon, Antoine Bosselut, Wen tau Yih, and Peter Clark. 2019. Everything happens for a reason: Discovering the purpose of actions in procedural text. In *EMNLP*.

Rajarshi Das, Tsendsuren Munkhdalai, Xingdi Yuan, Adam Trischler, and Andrew McCallum. 2019. Building dynamic knowledge graphs from text using machine reading comprehension. In *ICLR*.

Xinya Du, Bhavana Dalvi, Niket Tandon, Antoine Bosselut, Wen tau Yih, Peter Clark, and Claire Cardie. 2019a. Be consistent! improving procedural text comprehension using label consistency. In *NAACL*.

Xinya Du, Bhavana Dalvi, Niket Tandon, Antoine Bosselut, Wen tau Yih, Peter Clark, and Claire Cardie. 2019b. Improving procedural text comprehension using label consistency. In *NAACL*.

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

Shen Gao, Xiuying Chen, Zhaochun Ren, Dongyan Zhao, and Rui Yan. 2020. From standard summarization to new tasks and beyond: Summarization with manifold information. In *IJCAI*.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018a. Bottom-up abstractive summarization. In *EMNLP*.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018b. Bottom-up abstractive summarization. In *EMNLP*.

Saransh Goyal, P. Pandey, Garima Gaur, D Subhalingam, Srikanta J. Bedathur, and Maya Ramanath. 2021. Tracking entities in technical procedures - a new dataset and baselines. *ArXiv*, abs/2104.07378.

Aditya Gupta and Greg Durrett. 2019a. Effective use of transformer networks for entity tracking. In *EMNLP*.

Aditya Gupta and Greg Durrett. 2019b. Tracking discrete and continuous entity state for process understanding. In *Proceedings of the Third Workshop on Structured Prediction for NLP*, pages 7–12, Minneapolis, Minnesota. Association for Computational Linguistics.

Hanqi Jin and Xiaojun Wan. 2020. Abstractive multi-document summarization via joint learning with single-document summarization. In *EMNLP*.

Chloé Kiddon, Ganesa Thandavam Ponnuraj, Luke Zettlemoyer, and Yejin Choi. 2015. Mise en place: Unsupervised interpretation of instructional recipes. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 982–992, Lisbon, Portugal. Association for Computational Linguistics.

Mahnaz Koupaee and William Yang Wang. 2018. Wikihow: A large scale text summarization dataset. *ArXiv*, abs/1810.09305.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer.

2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*.

Rongzhong Lian, Min Xie, Fan Wang, Jinhua Peng, and Hua Wu. 2019. Learning to select knowledge for response generation in dialog systems. In *IJCAI*.

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

Junyang Lin, Xu Sun, Shuming Ma, and Qi Su. 2018. Global encoding for abstractive summarization. In *ACL*.

Yixin Liu and Pengfei Liu. 2021. Simcls: A simple framework for contrastive learning of abstractive summarization. In *ACL*.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *ACL*.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *EMNLP*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Alec Radford, Jeff Wu, R. Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. In *Arxiv*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Hannah Rashkin, Antoine Bosselut, Maarten Sap, Kevin Knight, and Yejin Choi. 2018. Modeling naive psychology of characters in simple commonsense stories. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2289–2299, Melbourne, Australia. 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*.

Y. Sun and Jiawei Han. 2013. Mining heterogeneous information networks: a structural analysis approach. *SIGKDD Explor.*, 14:20–28.

Y. Sun, Jiawei Han, X. Yan, Philip S. Yu, and Tianyi Wu. 2011. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proc. VLDB Endow.*, 4:992–1003.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Daivik Swarup, Ahsaas Bajaj, Sheshera Mysore, Tim O'Gorman, Rajarshi Das, and Andrew McCallum. 2020. An instance level approach for shallow semantic parsing in scientific procedural text. In *EMNLP*.

Niket Tandon, Bhavana Dalvi, Joel Grus, Wen tau Yih, Antoine Bosselut, and Peter Clark. 2018. Reasoning about actions and state changes by injecting commonsense knowledge. In *EMNLP*.

Niket Tandon, Keisuke Sakaguchi, Bhavana Dalvi, Dheeraj Rajagopal, Peter Clark, Michal Guerquin, Kyle Richardson, and Eduard Hovy. 2020. A dataset for tracking entities in open domain procedural text. In *EMNLP*.

Jizhi Tang, Yansong Feng, and Dongyan Zhao. 2020. Understanding procedural text using interactive entity networks. In *EMNLP*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *WWW*.

Hongming Zhang, Muhao Chen, Haoyu Wang, Yangqiu Song, and Dan Roth. 2020a. Analogous process structure induction for sub-event sequence prediction. In *EMNLP*.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020b. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *ICML*.

Li Zhang, Qing Lyu, and Chris Callison-Burch. 2020c. Reasoning about goals, steps, and temporal ordering with wikihow. In *EMNLP*.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *ACL*.

Chenguang Zhu, William Hinthorn, Ruochen Xu, Qingkai Zeng, Michael Zeng, Xuedong Huang, and Meng Jiang. 2021. Enhancing factual consistency of abstractive summarization. In *NAACL*.