Constructing Procedural Graphs with Multiple Dependency Relations: A New Dataset and Baseline

Haopeng Ren[1,2]*, **Yushi Zeng**[1,2]*, **Yi Cai**[1,2]†, **Bihan Zhou**[1,2], **Zetao Lian**[1,2]
[1]School of Software Engineering, South China University of Technology, Guangzhou, China
[2]Key Laboratory of Big Data and Intelligent Robot (South China University of Technology), Ministry of Education
ycai@scut.edu.cn

Abstract

Current structured and semi-structured knowledge bases mainly focus on representing descriptive knowledge but ignore another commonsense knowledge (Procedural Knowledge). To structure the procedural knowledge, existing methods are proposed to automatically generate flow graphs from procedural documents. They focus on extracting sequential dependency between sentences but neglect another two important dependencies (i.e., inclusion dependency and constraint dependency) in procedural documents. In our paper, we explore a problem of automatically generating procedural graph with multiple dependency relations to extend the flow graph constructed by existing methods and propose a procedural graph construction method with syntactic information and discourse structures. A new dataset (WHPG) is built and extensive experiments are conducted to evaluate the effectiveness of our proposed model.

1 Introduction

Many well-known structured knowledge bases (e.g., *Wikidata*[1]) and semi-structured knowledge bases (e.g., *Wikipedia*[2]) have been built and assist many applications to achieve remarkable performance, such as the question-answering (QA) (Li and Moens, 2022), information retrieval (Zhou et al., 2022)) and recommendation systems (Cui and Lee, 2022). They focus on representing the *descriptive knowledge* i.e. the knowledge of attributes or features of things (Yang and Nyberg, 2015), but lack another kind of commonsense knowledge—*Procedural Knowledge*. Specifically, the knowledge which is in the form of procedures or sequences of actions to achieve particular goals is called as procedural knowledge, such as cooking recipes and maintenance manuals.

Generally, most procedural knowledge is expressed in unstructured texts (e.g., websites or books of cooking recipes). To extract the structured procedural knowledge, existing methods (Honkisz et al., 2018; Qian et al., 2020; Pal et al., 2021) are designed to transform the unstructured procedural documents into flow graphs (or workflows) which can effectively present the main operations and their ordering relations expressed in procedural documents. However, they only focus on extracting the *sequential dependency* (i.e., the dependency relation "*Next*" in Figure 1) between steps (operational sentences) in a procedural document, which is insufficient in real-world scenarios. As shown in Figure 1, sentences $S2$ and $S3$ are the *sub-actions* of sentence $S1$, which provide more fine-grained operational statements to finish operation $S1$. There is another kind of dependency—*inclusion dependency* between sentences $S1$ and $S2$ (or between $S1$ and $S3$). Nevertheless, the flow graphs constructed by current methods (Qian et al., 2020; Pal et al., 2021) ignore the inclusion dependencies among sentences and wrongly connect sentences $S1$ and $S2$ as a "*Next*" relation, as shown in Figure 1.

Furthermore, declarative (or descriptive) sentences commonly appear in real-world procedural documents, which state the constraints (e.g, reasons, conditions and effects) of doing things. Current researches have shown that declarative sentences in procedural documents can provide important clues for the procedural semantic understanding and reasoning (Georgeff and Lansky, 1986) in many downstream tasks such as operation diagnosis (Luo et al., 2021) and technical maintenance (Hoffmann et al., 2022). However, current knowledge structure methods (Qian et al., 2020; Pal et al., 2021) simply transform the declarative sentences into an information flow in a flow graph (e.g., $S7 \rightarrow S8$ in Figure 1), which neglects the *constraint dependency* between operational and

*The authors contribute equally.
†Corresponding author
[1]http://wikidata.org/
[2]http://wikipedia.org/

Findings of the Association for Computational Linguistics: ACL 2023, pages 8474–8486
July 9-14, 2023

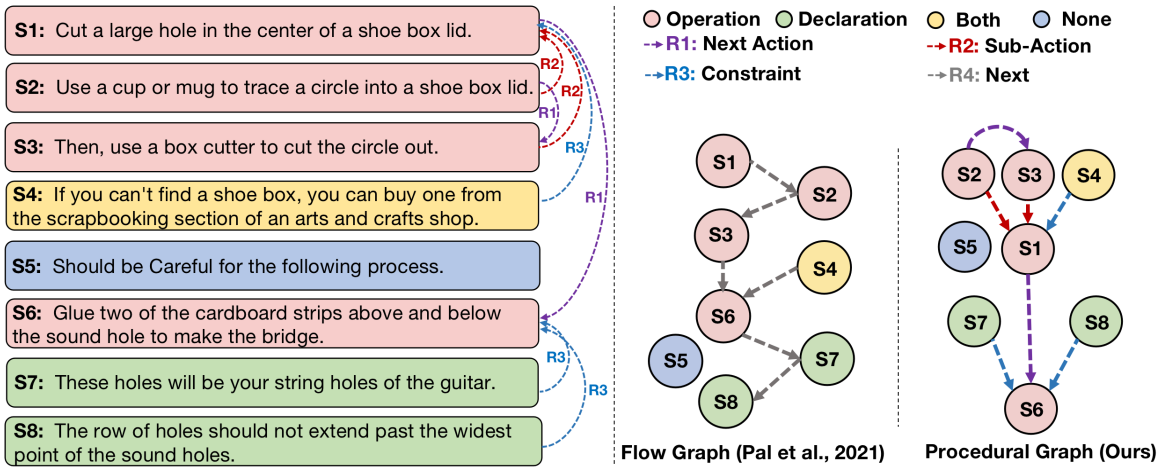

Figure 1: An Example of Flow Graph (Pal et al., 2021) and Procedural Graph (Ours) with Multiple Dependency Relations.

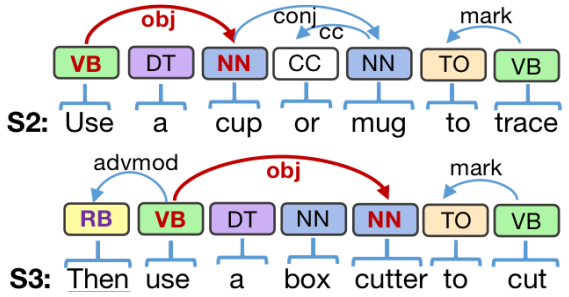

Figure 2: Syntactic Structures of Procedural Sentences.

declarative sentences. As shown in Figure 1, the declarative sentences $S7$ and $S8$ respectively describe the effect constraint and condition constraint for the execution of operational sentence $S6$.

Based on the above motivations, we explore a problem of automatically constructing a procedural graph with multiple dependency relations between sentences in a procedural document. According to our observation, the syntactic structures of sentences can provide obvious features for identifying sentence types and then assist to detect the dependency relations between sentences. As shown in Figure 2, the syntactic pattern "*verb(VB)* $\xrightarrow{obj}$ *noun(NN)*" is a strong indicator for classifying sentences $S2$ and $S3$ into *operation* types. Meanwhile, the sentence type prediction can further benefit the dependency relation detection between sentences. For example, the *constraint dependency* cannot exist between two sentences with an *operation* type. Moreover, inspired by researches in discourse parsing (Zhu et al., 2019; Wang et al., 2021), we observe that the contextual dependency structures (which is called as discourse structures) can provide features to recognize the dependency relations between sentences. As shown in Figure 1, the dependency relation between $S1$ and $S3$ can be inferred as *Sub-Action* according to their contextual dependency structure $S1 \overset{\text{Sub-Action}}{\longleftarrow} S2 \overset{\text{Next-Action}}{\longrightarrow} S3$.

In our paper, we design a procedural graph construction method to detect the multiple dependency relations between sentences in procedural documents by utilizing syntactic information and discourse structures. Specifically, a GCN-based syntactic structure encoder with multi-query attention is proposed to capture the syntactic features of sentences and improve the ability to distinguish between operational and declarative sentences. Moreover, a structure-aware edge encoder is designed to assist the inference of dependencies between sentences by infusing the contextual structure features in procedural documents. Furthermore, due to the lack of dependencies between sentences in existing procedural text datasets, a new dataset *WHPG* is built based on the wikiHow[3] database.

To summarize, the main contributions of this paper are listed as follows:

- We explore a problem of automatically generating procedural graphs with multiple dependencies from unstructured procedural documents, aiming to extend the flow graphs in existing methods that ignore dependencies of sentences. To the best of our knowledge, our work is the first study focusing on generating procedural graphs with multiple dependencies from procedural documents.

[3] http://www.wikihow.com/

- We design a GCN-based syntactic structure encoder and a discourse-structure aware edge encoder to effectively identify node types and assist the detection of dependency relations in procedural documents.
- We create a new procedural text dataset *WHPG* which builds dependency relations between operational and declarative sentences. Extensive experiments are conducted on two public datasets from different domains and our created dataset to evaluate the effectiveness of our model in automatically generating the procedural graph.

2 Related Work

Many prominent knowledge bases such as *WikiData* (Vrandečić and Krötzsch, 2014), *Wikipedia* (Lehmann et al., 2015) and *FreeBase* (Bollacker et al., 2007) mainly focus on representing *descriptive knowledge* (i.e., the knowledge of attributes or features of things (Yang and Nyberg, 2015)). But they do not sufficiently cover *procedural knowledge* (i.e., the knowledge of procedures or sequences of actions for achieving the particular goals (Georgeff and Lansky, 1986)). Recently, to obtain the structured procedural knowledge, two categories of methods (i.e., entity-level and sentence-level based methods) are proposed. Specifically, the entity-level based methods (Jermsurawong and Habash, 2015; Feng et al., 2018; Mysore et al., 2019; Qian et al., 2020; Xu et al., 2020; Yamakata et al., 2020; Jiang et al., 2020; Fang et al., 2022) aim to extract the predefined entities and their relations from unstructured procedural texts (e.g., cooking recipes). However, they require large-scale fine-grained annotated data for each domain and lack domain generalization ability.

To alleviate these issues, sentence-level based methods (Pal et al., 2021) are designed, aiming to construct the flow graphs at sentence-level for procedural documents. However, they only focus on extracting the action flows with sequential dependencies, which is limited in real-world scenarios. In practice, both the *inclusion dependency* and *constraint dependency* are common in procedural texts and benefit the procedural text understanding and reasoning (Georgeff and Lansky, 1986; Hoffmann et al., 2022). Thus, our paper explores a problem of automatically generating a procedural graph with dependency relations from a procedural document. Up to the present, several public entity-level procedural text datasets (Yamakata et al., 2020; Qian et al., 2020; Mysore et al., 2019; Mishra et al., 2018) and a sentence-level dataset *CTFW* (which is not publicly available due to ethical considerations) (Pal et al., 2021) are built. Nevertheless, existing public datasets do not annotate the dependency relations between sentences in a procedural document. Thus, a new dataset *WHPG* based on the *wikiHow* knowledge base is built and will be publicly available for evaluation in future research.

3 Model

3.1 Problem Definition and Notations

The goal of our task is to extract the dependency relations between sentences and construct a procedural graph for each procedural document. Specifically, given a procedural document $D = \{s_1, s_2, \ldots, s_N\}$, a procedural graph $G_D = \{D, \Psi, R\}$ with nodes (i.e., sentences) $s_i \in D$ and triplets $(s_i, r_{i,j}, s_j) \in \Psi$ is constructed, where $r_{i,j} \in R$ denotes the dependency relation between sentences s_i and s_j; N is the number of sentences in a procedural document D. Note that the dependency relation set R contains four kinds of dependency relations: *Next-Action*, *Sub-Action*, *Constraint* and *None*, as shown in Figure 1.

To construct the procedural graphs, three subtasks (i.e., *Node Type Classification*, *Edge Prediction* and *Dependency Relation Classification*) are required. For *Node Type Classification* task, each node $s_i \in D$ is classified into one of the node types (i.e., *Operation*, *Declaration*, *Both* and *None*). Then, the extraction of triplets $(s_i, r_{i,j}, s_j)$ from the procedural document D can be divided into two tasks: *Edge Prediction* $P(s_i \rightarrow s_j | (s_1, s_2, \ldots, s_N))$ which predicts whether an edge exists for each sentence pair S_i and S_j; and *Dependency Relation Classification* $P(r_{i,j} | s_i \rightarrow s_j)$ which classifies each sentence pair (predicted as an edge in *Edge Prediction* task) into one of the dependency relations (i.e., *Next-Action*, *Sub-Action* and *Constraint*).

3.2 Syntactic Graph Construction

The part-of-speech and syntactic structure of sentences can provide the evident clues for facilitating the inference of node types and dependency relations. For each sentence, we use the Standford CoreNLP libraries (Manning et al., 2014) to recognize the part-of-speech of each token and

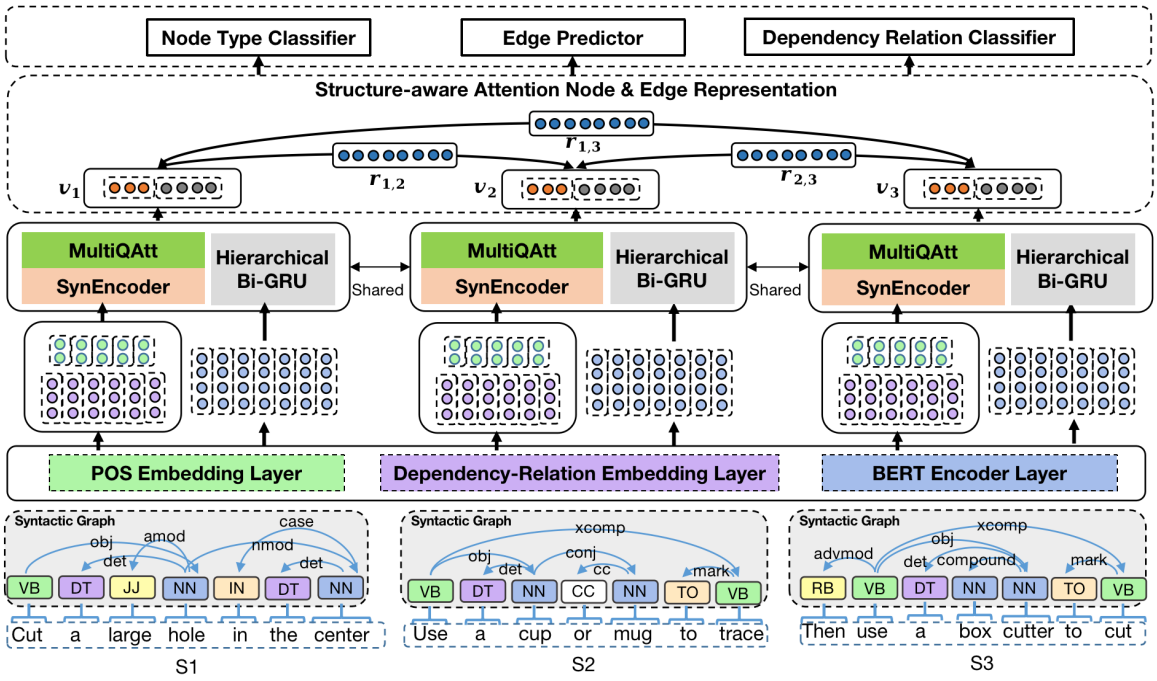

Figure 3: The Overview of Our Proposed Model with Syntactic and Document Structures.

dependency relations among tokens, as shown in Figure 3. Thus, given the sentence $s_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,n}\}$, a syntactic relational graph is created as follows:

$$G_i^{syn} = \{P_i, \Phi_i, R_i^{syn}\} \quad (1)$$

where P_i represents a set of the part-of-speech in sentence s_i; R_i^{syn} is a set of syntactic dependency relations and Φ_i denotes a set of triplets $(p_{i,j}, r_{j,k}^{syn}, p_{i,k})$ with the part-of-speech $p_{i,j} \in P_i$ of the token $x_{i,j}$ and syntactic dependency relation $r_{j,k}^{syn} \in R_i^{syn}$ between part-of-speech $p_{i,j}$ and $p_{i,k}$.

3.3 Sentence (Node) Feature Representation

3.3.1 Syntactic RGCN Encoder

Each type of part-of-speech $p \in P$ and dependency relation $r \in R^{syn}$ are respectively initialized into a learnable vector $\boldsymbol{p} \in \mathbb{R}^{d_r}$ and a learnable weight matrix $\boldsymbol{W}_r \in \mathbb{R}^{d_r \times d_r}$. Then, given a syntactic graph G_i^{syn} for sentence s_i, each part-of-speech node $p_{i,j}$ is encoded by Relational Graph Convolutional Networks (RGCN) (Schlichtkrull et al., 2018) encoder as follows:

$$\boldsymbol{p}_{i,j}^{(l+1)} = \sigma\Big(\sum_{r \in R_i^{syn}} \sum_{k \in N_j^r} \frac{1}{|N_j^r|} \boldsymbol{W}_r^{(l)} \boldsymbol{p}_{i,k}^{(l)} + \boldsymbol{W}_0^{(l)} \boldsymbol{p}_{i,j}^{(l)}\Big) \quad (2)$$

where N_j^r is the set of neighborhood node for the node $p_{i,j}$ with the relation $r \in R_i^{syn}$; $\boldsymbol{W}_0 \in \mathbb{R}^{d_r}$ denotes the learning parameters and l is the number of layers in RGCN encoder.

3.3.2 Multi-Query Syntactic-Aware Attention

Moreover, not all the syntactic features are equally important to identify the node types and the dependency relations between the nodes. For example shown in Figure 3, the syntactic pattern "*VB* $\xrightarrow{obj}$ *NN*" is a strong indicator for classifying nodes as "*Operation*" types, while the pattern "*Determiner (DT)* $\xrightarrow{det}$ *Noun (NN)*" does not provide explicit features for the node type classification. Meanwhile, different tasks also focus on different syntactic features. Motivated by this, a multi-query syntactic-aware attention module is designed to enable the model to pay attention to the relevant syntactic features for the target tasks. Specifically, given a sentence s_i, the syntactic feature representation is obtained as follows:

$$\boldsymbol{v}_i^{syn} = \sum_{k=1}^{N_q} \sum_{j=1}^{n} \alpha_{i,j}^k \boldsymbol{p}_{i,j}$$
$$\alpha_{i,j}^k = \frac{exp(\boldsymbol{q}_k \boldsymbol{p}_{i,j})}{\sum_{m=1}^{n} exp(\boldsymbol{q}_k \boldsymbol{p}_{i,m})} \quad (3)$$

where n is the number of tokens in sentence s_i; N_q is the number of query and $\boldsymbol{q}_k \in \mathbb{R}^{d_r}$ denotes a learnable vector.

3.3.3 BERT-based Bi-GRU Encoder

For a sentence s_i, each token $x_{i,j}$ can be encoded into a numerical vector $\boldsymbol{v}_{i,j} \in \mathbb{R}^{d_{bert}}$ by the pre-trained language model BERT (Kenton and Toutanova, 2019). Then, a hierarchical GRU encoder consisting of two bidirectional GRUs (Bi-GRU) is utilized to learn the contextual features. Specifically, given a sequence of embedding vectors $\boldsymbol{E}_i = \{\boldsymbol{v}_{i,1}, \boldsymbol{v}_{i,2}, \ldots, \boldsymbol{v}_{i,n}\}$ of sentence s_i, the first Bi-GRU is utilized to encode them as $\boldsymbol{h}_i \in \mathbb{R}^{d_{gru}}$ by concatenating the last hidden states from the two directions. In this way, each sentence s_i is encoded into a vector $\boldsymbol{h}_i$. Then, the procedural document D can be encoded as a sequence of vectors $\boldsymbol{H}^{sent} = \{\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_N\}$, where N is the number of sentences in the procedural document. Moreover, to capture the global contextual features, the second Bi-GRU encoder is adopted to transform $\boldsymbol{H}^{sent}$ into $\boldsymbol{H}^{dialog} = \{\boldsymbol{v}_1^{gru}, \boldsymbol{v}_2^{gru}, \ldots, \boldsymbol{v}_N^{gru}\}$, where $\boldsymbol{v}_i^{gru} \in \mathbb{R}^{d_{gru}}$ denotes the feature representation of the sentence $s_i \in D$.

3.3.4 Feature Fusion

Each sentence $s_i \in D$ can be obtained by concatenating the syntactic feature representation $\boldsymbol{v}_i^{syn}$ and the semantic feature representation $\boldsymbol{v}_i^{gru}$, as follows:

$$\boldsymbol{v}_i = [\boldsymbol{v}_i^{syn}; \boldsymbol{v}_i^{gru}] \tag{4}$$

where $[\cdot;\cdot]$ denotes the concatenation operation for the given two vectors.

3.4 Structural-Aware Edge Feature Representation

The contextual dependency structures in a procedural document have been proven to be effective in discourse parsing (Shi and Huang, 2019; Wang et al., 2021). The structure-aware attention is designed to capture the contextual structure features for each target sentence pair in both edge prediction and relation classification. Specifically, given a node pair (s_i, s_j), the edge representation $\boldsymbol{r}_{i,j}^{init}$ is initialized by concatenating the syntactic feature representation $\boldsymbol{v}_i^{syn}$, $\boldsymbol{v}_j^{syn}$ and the distance embedding $\boldsymbol{v}_{i,j}^{dist}$ as follows:

$$\boldsymbol{r}_{i,j}^{init} = \sigma([\boldsymbol{v}_i^{syn}; \boldsymbol{v}_{i,j}^{dist}; \boldsymbol{v}_j^{syn}]\boldsymbol{W}^C) \tag{5}$$

where $i < j$, $j - i < win$ and win is the longest distance length between the given two nodes in a procedural document. Then, we update the node representation $\boldsymbol{v}_i$ in Equation (4) with the contextual features as follows:

$$\begin{aligned} \boldsymbol{v}_i^{att} &= \sum_{j=1}^{N} \alpha_{i,j}(\boldsymbol{v}_j\boldsymbol{W}^V + \boldsymbol{r}_{i,j}^{init}\boldsymbol{W}^F) \\ \alpha_{i,j} &= \frac{exp(e_{i,j})}{\sum_{k=1}^{N} exp(e_{i,k})} \\ e_{i,j} &= \frac{(\boldsymbol{v}_i\boldsymbol{W}^Q)(\boldsymbol{v}_j\boldsymbol{W}^K + \boldsymbol{r}_{i,j}^{init}\boldsymbol{W}^R)^T}{\sqrt{d_r + d_{gru}}} \end{aligned} \tag{6}$$

where $\boldsymbol{W}^Q$,$\boldsymbol{W}^F$,$\boldsymbol{W}^K$, $\boldsymbol{W}^V$ and $\boldsymbol{W}^R$ are learnable parameters and $d_r + d_{gru}$ is the dimension of the node representations. Finally, the edge representation $\boldsymbol{r}_{i,j}$ is calculated by refusing the node features, as follows:

$$\begin{aligned} \gamma_{i,j} &= \sigma([\boldsymbol{v}_i^{att}; \boldsymbol{v}_j^{att}]\boldsymbol{W}^r) \\ z_{i,j} &= \sigma([\boldsymbol{v}_i^{att}; \boldsymbol{v}_j^{att}]\boldsymbol{W}^z) \\ \boldsymbol{r}_{i,j}^{'} &= tanh([\gamma_{i,j} \odot \boldsymbol{r}_{i,j}^{init}; \boldsymbol{v}_i^{att}; \boldsymbol{v}_j^{att}]\boldsymbol{W}^h) \\ \boldsymbol{r}_{i,j} &= (1 - z_{i,j}) \odot \boldsymbol{r}_{i,j}^{init} + z_{i,j} \odot \boldsymbol{r}_{i,j}^{'} \end{aligned} \tag{7}$$

where $\boldsymbol{W}^r$, $\boldsymbol{W}^z$ and $\boldsymbol{W}^h$ are the learnable parameters; $\odot$ denotes the dot-product operation.

3.5 Projection and Loss Function

The representation of node s_i and edge $r_{i,j}$ can be encoded by Equation (4) and Equation (7) as $\boldsymbol{v}_i$ and $\boldsymbol{r}_{i,j}$. We adopt a projection layer with a softmax function to calculate the probability distribution of categories (i.e., {*Operation, Declaration, Both, None*} for node type classification task; {*Existing, Non-Existing*} for edge prediction task and {*Next-Action, Sub-Action, Constraint, None*} for dependency relation classification task).

Given the training dataset M, the model is trained with the following training objective:

$$\mathcal{L}(M, \theta) = \sum_{D \in M} (\mathcal{L}_t(D; \theta) + \mathcal{L}_e(D; \theta) + \mathcal{L}_r(D; \theta)) \tag{8}$$

where $\mathcal{L}_t(D; \theta)$, $\mathcal{L}_e(D; \theta)$ and $\mathcal{L}_r(D; \theta)$ are the cross-entropy loss functions for *node type classification*, *edge prediction* and *relation classification* tasks; and D is a procedural document from the training dataset M.

4 Experiment

We firstly introduce the construction of the new dataset *WHPG* and then analyze the experimental results in detail.

Dataset statistics	COR	MAM	CTFW	**WHPG**
# Doc.	297	575	3154	**283**
Avg Size of Doc.	9.52	8.12	17.11	**23.47**
Avg Len. of Sent.	65.46	34.81	92.87	**79.76**
# Edges ($\|e^+\|$)	2670	5043	54539	**7341**
$\|e^+\| : (\|e^+\| + \|e^-\|)$	0.18	0.12	0.07	**0.07**
Avg degree of node	1.83	1.76	1.88	**2.21**

Table 1: Dataset Statistics. $|e^+| + |e^-|$ is the total number of actual edges $|e^+|$ and possible edges $|e^-|$.

Node Type	Size
Operation	3585
Declaration	1794
Both	1100
None	163

Relation Type	Size
Next-Action	2272
Sub-Action	2371
Constraint	2698

Table 2: The Size of Sentence (Node) Types and Dependency Relations in *WHPG*.

Label	Train	Validation	Test
Operation	2471	412	702
Declaration	1261	163	370
Both	743	125	232
None	139	18	6

Table 3: Label Distributions of Sentence (Node) Type Classification.

Label	Train	Val	Test
Next-Action	1542	264	466
Sub-Action	1630	270	471
Constraint	1881	280	537

Table 4: Label Distributions of Dependency Relation Classification

4.1 Dataset Collection & Annotation

We build the original corpus from the online wikiHow knowledge base (Anthonio et al., 2020) which provides a collection of *how-to* articles about various topics (e.g., entertainment and crafts). We exploit the wikiHow knowledge base to create *WHPG*, a dataset of procedural texts with dependency-relation annotations among operational and declarative sentences. The online wikiHow knowledge base provides an *Export pages*[4] service which allows exporting the texts of wikiHow articles (Anthonio et al., 2020). We adopt the python library *urllib*[5] to request the Export pages services and crawl procedural articles. With the candidate set of procedural documents, we filter out the unnecessary information (e.g., writing date, citations and URLs). The procedural documents containing only one step are also filtered out. Finally, three parts (i.e., titles, method names and steps of procedural documents) are kept to form a complete instance. Statistically, we obtain a candidate set of 330 procedural documents from the *Crafts* topic.

As shown in Figure 1, for each procedural document, we provide three kinds of annotations: *sentence type* (i.e., "*Operation*", "*Declaration*", "*Both*" or "*None*"), *edge* (i.e., the connections between two sentences) and *dependency relation* (i.e., "*Next-Action*", "*Sub-Action*" and "*Constraint*"). Three well-educated annotators are employed to make annotations by averaging the candidate procedural documents using the BRAT tool[6]. To ensure the annotation quality, each annotator are required to give the confidence score for each annotated label. We weigh the confidence score of each annotator for the same label and the label with the highest score will be preserved. Moreover, the annotation samples with the lowest confidence scores will be brainstormed to determine the final annotation results. Moreover, the labeled instances that are difficult to reach a consensus will be discarded. Finally, two well-trained annotators are required to recheck all the annotation results to further ensure the annotation quality.

Finally, the final dataset contains 283 procedural documents with about 7341 edges. The statistical comparison of WHPG with existing sentence-level procedural text datasets is shown in Table 1. Moreover, the statistics of sentence (node) types and dependency relations of our created dataset *WHPG* is shown in Table 2. We also show the label distributions of the node types and dependency relations, as shown in Table 3 and Table 4.

4.2 Experiment Settings

4.2.1 Datasets & Experimental Settings

We conduct extensive experiments[7] on our annotated dataset *WHPG*. Following Pal et al. (2021), we split *WHPG* dataset into train, validation and test sets with 7:1:2 ratio. Furthermore, two public datasets (i.e., *COR* (Yamakata et al., 2020) and MAM (Qian et al., 2020) which do not consider dependency relations among sentences) are also

[4] https://www.wikihow.com/index.php?title=Special:Export&action=submit

[5] https://docs.python.org/2/library/urllib2.html

[6] http://brat.nlplab.org/index.html

[7] The code and datasets are publicly available at https://github.com/betterAndTogether/WHPG

Settings	$win = 5$		$win = 10$		$win = 20$		ALL	
	Edge	Edge&Rel	Edge	Edge&Rel	Edge	Edge&Rel	Edge	Edge&Rel
BERT-NS	55.44	22.98	39.24	16.53	28.05	11.16	21.28	10.02
RoBERTa-NS	55.54	23.07	40.02	16.31	27.82	11.29	21.74	8.43
BERT-GCN	42.57	17.69	27.32	11.73	20.57	8.28	18.02	7.24
RoBERTa-GCN	42.59	16.75	27.23	10.87	18.65	7.99	16.20	6.32
BERT-GAT	49.07	21.99	32.91	13.76	23.29	9.84	16.83	6.52
RoBERTa-GAT	47.34	19.87	34.13	14.69	22.48	9.12	18.60	7.32
BERT+SBil	-	-	-	-	-	-	29.67	17.58
Ours w/o SynEncoder	62.63	41.85	60.16	39.38	59.13	38.80	57.36	37.85
Ours w/o MultiQAtt	65.01	42.98	62.94	40.52	60.83	39.47	59.22	38.54
Ours w/o SAtt	59.52	38.98	58.72	37.71	57.40	36.67	55.47	35.43
Ours	**65.71**	**43.61**	**63.79**	**41.39**	**61.87**	**40.31**	**60.84**	**39.06**

Table 5: The Experiment Results (F1 scores) on *WHPG* dataset. *Edge* denotes the performance of edge prediction task only, and *Edge&Rel* which is the **main** metric, denotes the performance when both edge prediction and dependency relation classification tasks are correctly predicted at the same time. *SynEncoder* denotes the *Syntactic RGCN Encoder*; *MultiQAtt* denotes the *Multi-Query Syntactic-Aware Attention* module; *SAtt* denotes the *Structural-Aware Attention* module.

utilized to conduct the comparative experiments on the *Edge Prediction* task.

For the *Node Type Classification* task, we use the *accuracy* as the evaluation metric. Considering the label imbalance in *Edge Prediction* task, *F1-score* of the positive class (i.e., the sentence pair existing an edge) is used as the evaluation metric for the *Edge Prediction*. The performance of *Dependency Relation classification* task is affected by the previous stage *Edge Prediction*. Thus, we combine them to make an evaluation with the *F1-score* metric (i.e., *Edge&Rel* in Table 5).

In the edge prediction and dependency relation classification tasks, each sentence in the procedural document needs to be respectively combined with all the following sentences to determine whether there is an edge and what types of dependency relations they have. To evaluate the generalization ability, four experimental settings (i.e., $win = 5$, $win = 10$, $win = 20$ and ALL) are used to evaluate the effectiveness of our proposed model. For example, for the $win = 5$ setting, given the first sentence $s_1 \in D$, five candidate sentence pairs i.e. $\{(s_1, s_2), (s_1, s_3), (s_1, s_4), (s_1, s_5), (s_1, s_6)\}$ should be examined by the model to predict whether there are edges and which type of dependency relation they belong to. In training stage, we use AdamW optimizer with 4 batch size, 2e-5 learning rate and 0.4 dropout rate.

4.3 Result Analysis

To evaluate the effectiveness of our proposed model on the three tasks (i.e., *Node Type Classification*, *Edge Prediction* and *Dependency Relation Classification*), we compare the performance of our proposed model with 7 recent related works (Pal et al., 2021; Zhou and Feng, 2022) which focus on constructing flow graphs from a procedural document, as shown in Table 5. To explore the problem of generating procedural graphs from procedural documents, a new dataset *WHPG* is built and utilized to perform comparative experiments on both edge prediction and dependency relation classification tasks. Moreover, another two public datasets (i.e., *COR* (Yamakata et al., 2020) and *MAM* (Qian et al., 2020)) from different domains are used to conduct the comparative experiments. Since these two public datasets ignore the dependencies between sentences, we can only perform experiments on *Edge Prediction* task.

4.3.1 Node Type Classification

As shown in Table 6, five baselines (Pal et al., 2021; Zhou and Feng, 2022) are used to perform the comparative experiments. Compared with them, syntactic structures are embedded into the node feature representations in our model. From the experimental results, our model achieves the highest accuracies than current related works on both validation and test datasets. It can evaluate that syntactic structure features can be effectively captured and further improve the ability to distinguish between operational and declarative sentences.

4.3.2 Edge & Relation Classification

Table 5 shows the comparative experimental results under four window size settings (i.e., 5, 10, 20 and ALL) on both edge prediction and dependency relation classification tasks. Our proposed

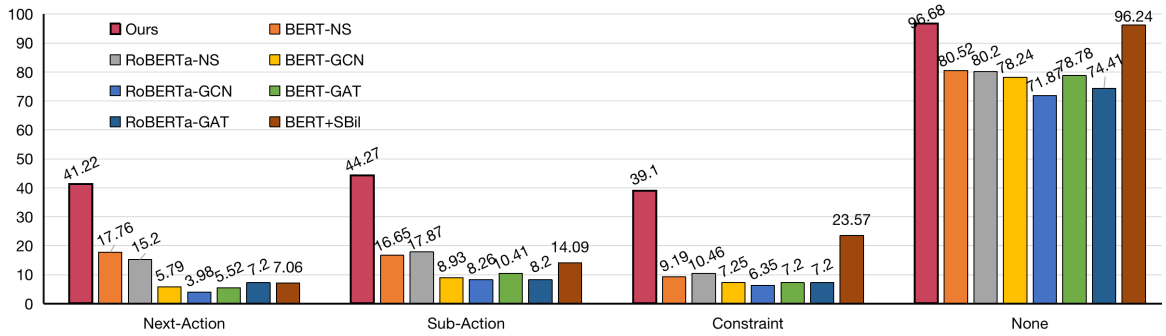

Figure 4: The Experimental Results (F1 scores) of Each Relation on *WHPG* Dataset. The results are reported when both edge prediction and relation classification tasks are correctly predicted at the same time.

Model	Val	Test
BERT-Base	76.04	80.76
RoBERTa-Base	75.06	80.45
BERT-Large	75.11	80.01
RoBERTa-Large	76.10	80.55
BERT+SBil	55.91	60.71
Ours	**76.90**	**81.38**

Table 6: Experimental Results (Mean Accuracy (%) from three seed values) of Node Type Classification Task on Both Validation and Test Set of *WHPG* dataset.

Model	COR	MAM
BERT-NS	43.14	29.73
RoBERTa	42.99	39.65
RoBERTa	42.99	39.65
BERT-GCN	58.13	63.75
RoBERTa-GCN	61.44	65.73
BERT-GAT	41.93	62.18
RoBERTa-GAT	24.74	59.55
BERT+SBil	46.76	58.21
Ours	**69.57**	**67.58**

Table 7: The Experimental Results (F1 scores) of Edge Prediction Task on *COR* and *MAM* Datasets.

model achieves the highest F1 scores with a large margin (nearly 15% F1 score in *Edge* and 20% F1 score in *Edge&Rel*) than current related works on all experiment settings. Specifically, current existing methods mainly focus on constructing a flow graph with only sequential dependency for each procedural document. They ignore another two significant dependencies (i.e., *inclusion dependency* and *constraint dependency*) between sentences. By capturing syntactic structures and discourse structures, our model can effectively identify the dependencies between sentences. Moreover, we observe that the performance of all models degrades as the window size increases (e.g., *BERT-NS* and *BERT-GCN* drop nearly 15% F1 score from 5 to 10 window size settings). Instead, our proposed model has the smallest performance drop (only nearly 3% F1 score) on both edge prediction and dependency relation classification when the window size is reduced. Comparing with current related works, the contextual dependency structure (discourse structure) features are utilized to assist the inference of detecting dependency relations between sentences. The experimental results can evaluate the advantages of our model with structure-aware attention module in handling long-range inter-sentence dependency recognition.

Furthermore, to evaluate the domain generalization ability of our model, another two public datasets (i.e., *COR* in recipe domain and *MAM* in the maintenance domain) are used to conduct the comparative experiments. As shown in Table 7, our model obtains the best performance with a large margin than all related works on both datasets. The experimental results evaluate that our proposed model can effectively identify sequential dependency between sentences and have a better domain generalization ability.

4.3.3 Analysis for Each Dependency Relation

Figure 4 shows the comparative experiments on extracting each type of dependency relations between sentences. Compared with the related works, our model can obtain the highest F1 scores on all dependency relation types. Note that due to the imbalance in the number of *existing* and *non-existing* edges between sentences in procedural documents, current existing methods are prone to recognize the inter-sentences as *None* dependency and have low performance in the three dependency relations (i.e., *Next-Action*, *Sub-Action* and *Constraint*).

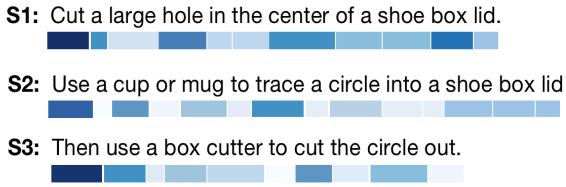

Figure 5: The heatmap of the weight distributions for each word measured by *Multi-Query Syntactic-Aware Attention*.

4.3.4 Ablation Study

As shown in Table 5, the ablation experiments are conducted to evaluate the effectiveness of the designed modules (i.e., *SynEncoder*, *MultiQAtt* and *SAtt*). The ablation experimental results can evaluate that both the syntactic information and discourse structure benefit the dependency relation detection. Specifically, both the *SynEncoder* and *MultiQAtt* modules can effectively capture the syntactic features and assist the dependency relation detection. Moreover, the performance of our model can be improved effectively when the discourse-structure features are embedded by the structure aware attention module.

4.4 Visualization

As shown in Figure 5, we show the weight distributions of each word measured by the *Multi-Query Syntactic Aware Attention Module*. We can observe that the phrases with the syntactic pattern "$VB \xrightarrow{obj} NN$" (e.g,, "cut→hole" in *S1* and "use→cup" in *S2*) obtain the higher weight values than other words, which indicates the sentences as operational sentences. Moreover, the token "*Then*" in *S3* of Figure 5 is measured as the highest weight value, which indicates the sentence has the "*Next-Action*" dependency relation with previous sentences. The visualization analyse can evaluate the effectiveness of our proposed module *Multi-Query Syntactic Aware Attention*.

5 Conclusion

In this paper, we explore a problem of automatically generating procedural graphs with multiple dependency relations for procedural documents. Existing procedural knowledge structured methods mainly focus on constructing action flows with the sequential dependency from procedural texts but neglect another two important dependencies: inclusion dependency and constraint dependency which are helpful for the procedural text understanding and reasoning. To solve this problem, we build a new procedural text dataset with multiple dependency relations and propose a procedural graph construction method by utilizing syntactic and discourse structure features. Extensive experiments are conducted and evaluate the effectiveness of our proposed model.

6 Limitations

In this section, we draw conclusions for the limitations of our proposed model in this paper. Our proposed model mainly focuses on the sentence-level procedural graph construction. The scenario that two actions in the same sentence cannot be considered in our proposed model. It is challenging to handle multi-grained (i.e., entity-level and sentence-level) dependencies between actions. We will consider this limitation as our future work.

Acknowledgement

This work was supported by the National Natural Science Foundation of China (62076100), Fundamental Research Funds for the Central Universities, SCUT (x2rjD2220050), the Science and Technology Planning Project of Guangdong Province (2020B0101100002), CAAI-Huawei MindSpore Open Fund, CCF-Zhipu AI Large Model Fund.

References

Talita Anthonio, Irshad Bhat, and Michael Roth. 2020. wikihowtoimprove: A resource and analyses on edits in instructional texts. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5721–5729.

Kurt Bollacker, Robert Cook, and Patrick Tufts. 2007. Freebase: A shared database of structured general human knowledge. In *AAAI*, volume 7, pages 1962–1963.

Limeng Cui and Dongwon Lee. 2022. Ketch: Knowledge graph enhanced thread recommendation in healthcare forums. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 492–501.

Biaoyan Fang, Timothy Baldwin, and Karin Verspoor. 2022. What does it take to bake a cake? the reciperef corpus and anaphora resolution in procedural text. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3481–3495.

Wenfeng Feng, Hankz Hankui Zhuo, and Subbarao Kambhampati. 2018. Extracting action sequences from texts based on deep reinforcement learning. In

Proceedings of the 27th International Joint Conference on Artificial Intelligence, pages 4064–4070.

Michael P Georgeff and Amy L Lansky. 1986. Procedural knowledge. *Proceedings of the IEEE*, 74(10):1383–1398.

Clemens Hoffmann, Sebastian Büttner, and Michael Prilla. 2022. Conveying procedural and descriptive knowledge with augmented reality. In *Proceedings of the 15th International Conference on PErvasive Technologies Related to Assistive Environments*, pages 40–49.

Krzysztof Honkisz, Krzysztof Kluza, and Piotr Wiśniewski. 2018. A concept for generating business process models from natural language description. In *International Conference on Knowledge Science, Engineering and Management*, pages 91–103. Springer.

Jermsak Jermsurawong and Nizar Habash. 2015. Predicting the structure of cooking recipes. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 781–786.

Yiwei Jiang, Klim Zaporojets, Johannes Deleu, Thomas Demeester, and Chris Develder. 2020. Recipe instruction semantics corpus (risec): Resolving semantic structure and zero anaphora in recipes. In *AACL-IJCNLP 2020, the 1st Conference of the Asia-Pacific Chapter of the Association Computational Linguistics and 10th International Joint Conference on Natural Language Processing*, pages 821–826. Association for Computational Linguistics (ACL).

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 6(2):167–195.

Mingxiao Li and Marie-Francine Moens. 2022. Dynamic key-value memory enhanced multi-step graph reasoning for knowledge-based visual question answering.

Ruipu Luo, Qi Zhu, Qin Chen, Siyuan Wang, Zhongyu Wei, Weijian Sun, and Shuang Tang. 2021. Operation diagnosis on procedure graph: The task and dataset. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 3288–3292.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Bhavana Dalvi Mishra, 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. *arXiv preprint arXiv:1805.06975*.

Sheshera Mysore, Zachary Jensen, Edward Kim, Kevin Huang, Haw-Shiuan Chang, Emma Strubell, Jeffrey Flanigan, Andrew McCallum, and Elsa Olivetti. 2019. The materials science procedural text corpus: Annotating materials synthesis procedures with shallow semantic structures. In *Proceedings of the 13th Linguistic Annotation Workshop*, pages 56–64.

Kuntal Kumar Pal, Kazuaki Kashihara, Pratyay Banerjee, Swaroop Mishra, Ruoyu Wang, and Chitta Baral. 2021. Constructing flow graphs from procedural cybersecurity texts. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3945–3957.

Chen Qian, Lijie Wen, Akhil Kumar, Leilei Lin, Li Lin, Zan Zong, Shu'ang Li, and Jianmin Wang. 2020. An approach for process model extraction by multi-grained text classification. In *International Conference on Advanced Information Systems Engineering*, pages 268–282. Springer.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer.

Zhouxing Shi and Minlie Huang. 2019. A deep sequential model for discourse parsing on multi-party dialogues. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7007–7014.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Ante Wang, Linfeng Song, Hui Jiang, Shaopeng Lai, Junfeng Yao, Min Zhang, and Jinsong Su. 2021. A structure self-aware model for discourse parsing on multi-party dialogues. In *IJCAI*, pages 3943–3949.

Frank F Xu, Lei Ji, Botian Shi, Junyi Du, Graham Neubig, Yonatan Bisk, and Nan Duan. 2020. A benchmark for structured procedural knowledge extraction from cooking videos. In *Proceedings of the First International Workshop on Natural Language Processing Beyond Text*, pages 30–40.

Yoko Yamakata, Shinsuke Mori, and John A Carroll. 2020. English recipe flow graph corpus. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5187–5194.

Zi Yang and Eric Nyberg. 2015. Leveraging procedural knowledge for task-oriented search. In *Proceedings of the 38th International ACM SIGIR Conference on*

Research and Development in Information Retrieval, pages 513–522.

Yifei Zhou and Yansong Feng. 2022. Improve discourse dependency parsing with contextualized representations. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2250–2261, Seattle, United States. Association for Computational Linguistics.

Ying Zhou, Xuanang Chen, Ben He, Zheng Ye, and Le Sun. 2022. Re-thinking knowledge graph completion evaluation from an informationd retrieval perspective. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 916–926.

Jie Zhu, Junhui Li, Muhua Zhu, Longhua Qian, Min Zhang, and Guodong Zhou. 2019. Modeling graph structure in transformer for better amr-to-text generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5459–5468.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
6

☐ A2. Did you discuss any potential risks of your work?
Not applicable. Left blank.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☒ Did you use or create scientific artifacts?

Left blank.

☐ B1. Did you cite the creators of artifacts you used?
No response.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
No response.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
No response.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
No response.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
No response.

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
No response.

C ☑ Did you run computational experiments?

Left blank.

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
4

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
4

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
4

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
4

D ☑ Did you use human annotators (e.g., crowdworkers) or research with human participants?
4

☑ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
4

☑ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
4

☑ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
4

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. 4

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. Left blank.