Procedural Text Understanding via Scene-Wise Evolution

Jialong Tang[1,3], **Hongyu Lin**[1], **Meng Liao**[4,*], **Yaojie Lu**[1,3],
Xianpei Han[1,2], **Le Sun**[1,2,*], **Weijian Xie**[4], **Jin Xu**[4]
[1] Chinese Information Processing Laboratory, Beijing, China
[2] State Key Laboratory of Computer Science Institute of Software, Chinese Academy of Sciences, Beijing, China
[3] University of Chinese Academy of Sciences, Beijing, China
[4] Data Quality Team, WeChat, Tencent Inc., China
{jialong2019,hongyu,yaojie2017,xianpei,sunle}@iscas.ac.cn
{maricoliao, vikoxie, jinxxu}@tencent.com

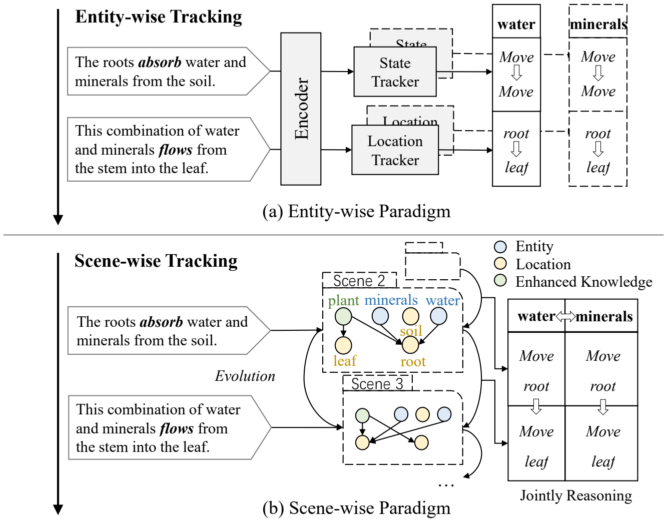
Abstract

Procedural text understanding requires machines to reason about entity states within the dynamical narratives. Current procedural text understanding approaches are commonly **entity-wise**, which separately track each entity and independently predict different states of each entity. Such an entity-wise paradigm does not consider the interaction between entities and their states. In this paper, we propose a new **scene-wise** paradigm for procedural text understanding, which jointly tracks states of all entities in a scene-by-scene manner. Based on this paradigm, we propose **S**cene **G**raph **R**easoner (**SGR**), which introduces a series of dynamically evolving scene graphs to jointly formulate the evolution of entities, states and their associations throughout the narrative. In this way, the deep interactions between all entities and states can be jointly captured and simultaneously derived from scene graphs. Experiments show that SGR not only achieves the new state-of-the-art performance but also significantly accelerates the speed of reasoning.

Introduction

Understanding how events will affect the world is the essence of intelligence (Henaff et al. 2017). Procedural text understanding, aiming to track the state changes (e.g., create, move, destroy) and locations (a span in the text) of entities throughout the whole procedure, is a representative task to estimate the machine intelligence on such ability (Mishra et al. 2018). For example, in Figure 1 (a), given a narrative describing the procedure of photosynthesis, as well as a pre-specified entity "*water*", a procedural text understanding model is asked to predict the corresponding {*State*, *location*} sequences: {*Move*, *root*}, {*Move*, *leaf*}. Compared with conventional factoid-style reading comprehension tasks (Seo et al. 2017; Clark and Gardner 2018), procedural text understanding is more challenging because it requires to model and reason with the dynamical world (Mishra et al. 2018; Bosselut et al. 2018).

Most approaches resolve procedural text understanding task in an **entity-wise** paradigm, where each entity is tracked separately, and state changes and locations of each entity are independently predicted. Along this line, as Figure 1 (a) shows, current procedural text understanding models mainly resort to hierarchical neural network architectures, which first encode the document-entity pair using a token-level encoder, then track the state changes and locations by two separate sentence-level trackers (Mishra et al. 2018; Du et al. 2019a,b; Tang, Feng, and Zhao 2020; Gupta and Durrett 2019b). More recently, the main research hot spot in this direction is how to obtain more effective document-entity representations by introducing graph-based architectures (Das et al. 2019; Zhong et al. 2020; Huang et al. 2021), pretrained language models (Gupta and Durrett 2019a; Amini et al. 2020; Zhang et al. 2020) or external knowledge bases (Ribeiro et al. 2019; Tandon et al. 2018).



(a) Entity-wise Paradigm

(b) Scene-wise Paradigm

Figure 1: Comparison between the traditional entity-wise paradigm and the proposed scene-wise paradigm for procedural text understanding. We can see that: (a) the entity-wise paradigm tracks each entity separately, and predict state changes and locations of each entity independently; (b) the scene-wise paradigm jointly tracks the state changes and locations of all entities scene-by-scene.

Unfortunately, the traditional entity-wise paradigm ignores the interactions between different entities in the same narrative, as well as the associations between the state

[*] Corresponding authors.

changes and locations of one entity. Specifically, the multiple entities mentioned in the same narrative are highly correlated with each other. For example, if we know "*water*" and "*minerals*" will be combined into a "*mixture*", we can confirm that they must in the same location "*leaf*". Furthermore, the states and locations of an entity are highly associated. For example, if we know the location of "*water*" changes from "*root*" to "*leaf*", we can easily predict the state of "*water*" is "*Move*". Besides, the states/locations at current step depend on the states/location at previous steps. For example, if we know "*root*" and "*leaf*" are parts of "*plant*", we will tend to predict the location "*leaf*" after the location "*root*". However, current entity-wise paradigm is unable to exploit the above-mentioned interactions and associations. In addition, reasoning procedures entity-by-entity is time-intensive and inefficient. Therefore, it is still far from achieving decent procedure text understanding models in both accuracy and efficiency.

To this end, this paper proposes **scene-wise** procedural text understanding, a new paradigm that jointly tracks the state changes and locations of all entities scene-by-scene. Instead of the entity-wise paradigm, we formulate the world described in the procedural text at different timesteps using a sequence of dynamically evolving scenes[1]. Figure 1 (b) illustrates the whole process of the scene-wise procedural text understanding. Specifically, each scene contains concepts (e.g., entities, locations or elements from external knowledge) and their relations at current timestep. As the narrative develops, the concepts and relations in the scene are dynamically evolved scene-by-scene. In this way, the state changes and locations of all entities are jointly exploited and then can be simultaneously derived from the scenes.

Based on this paradigm, we propose **S**cene **G**raph **R**easoner (**SGR**), a specific implementation for scene-wise procedural text understanding. SGR uses a graph structure to model scene. Each node in the graph represents a concept, and each edge in the graph represents a relation between two concepts. Then the scene evolution is modeled by the graph evolution throughout the whole procedure. Specifically, **SGR** consists of four basic components: 1) **a graph structure encoder**, which summarizes critical information from the current scene graph; 2) **a context encoder**, which captures the new events occurring from the sentence describing next narrative timestep; 3) **a graph structure predictor**, which predicts the evolution of the scene graph after the new events occurring; 4) **a state reasoner**, which distills the state changes and locations via comparing the adjacent scene graphs. By jointly exploiting all concepts and their relations in the scene graphs, SGR is able to better capture their interactions and associations throughout the whole procedure, and therfore enables to track the state changes and locations of all entities simultaneously in a graph evolution process.

Generally, the main contributions of this paper are:

- We propose a new scene-wise paradigm for procedural text understanding, which jointly tracks the state changes and locations of all entities scene-by-scene.
- We design a specific implementation SGR for scene-wise procedural text understanding, which can fully consider the interactions of multiple entities, as well as the associations of state changes and locations.
- We conduct experiments on ProPara (Mishra et al. 2018) and Recipes (Bosselut et al. 2018), two of the representative procedural text understanding benchmarks. Experiments show that SGR in the scene-wise paradigm achieves the new state-of-the-art procedural text understanding performance, and the reasoning speed is significantly accelerated.

Backgrounds

Task Definition

In this paper, we focus on ProPara (Mishra et al. 2018), which includes a variety of natural procedures, and the task is to answer the questions about the state changes and locations of the entities. Specifically, given:

- A paragraph P consists of T sentences $\{S_1, S_2, ..., S_T\}$;
- A set of pre-specified entities $E = \{e_1, e_2, ..., e_N\}$ need to be tracked;

the procedural text understanding model is required to reason with the described world, and output:

- State change sequences $Y^s = \{Y^s_{e_1}, Y^s_{e_2}, ..., Y^s_{e_N}\}$ for all pre-specified entities E, where $Y^s_{e_i} = \{y^s_{e_i,1}, y^s_{e_i,2}, ..., y^s_{e_i,T}\}$, $y^s_{e_i,t} \in \{$*Other (O)*, *Exist (E)*, *Move (M)*, *Create (C)*, *Destroy (D)*$\}$[2].
- Location sequences $Y^l = \{Y^l_{e_1}, Y^l_{e_2}, ..., Y^l_{e_N}\}$ for all pre-specified entities E, where $Y^l_{e_i} = \{y^l_{e_i,1}, y^l_{e_i,2}, ..., y^l_{e_i,T}\}$, $y^l_{e_i,t}$ is a text span in the paragraph. A special "*?*" token indicates the location is unknown.

Entity Recognition and Location Candidates Generation

In procedural text understanding, identifying entities is necessary because they are participants in the narrative. Thus, we first use SpaCy to tokenize the paragraph and all entities. All text are cleaned and lower-cased. And then the simple string matching algorithm is used to recognize entities.

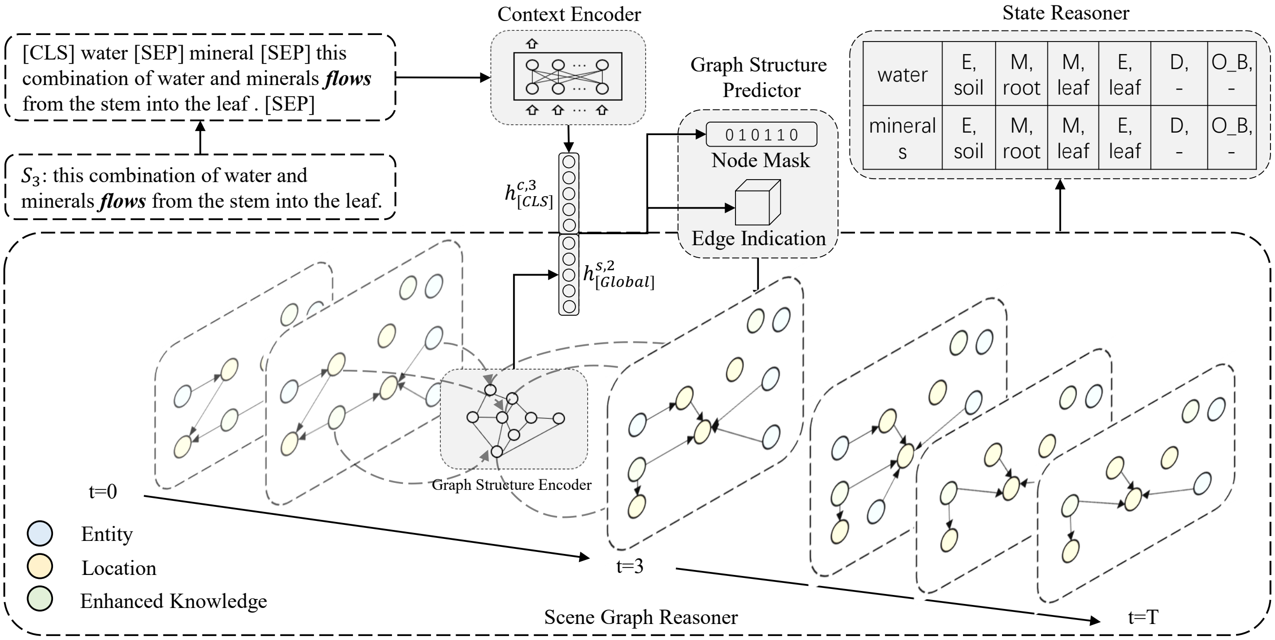
Unlike entities, location information in this task is not given initially. Due to the difficulty to consider arbitrary text spans as possible locations, we follow the previous works (Gupta and Durrett 2019b; Zhang et al. 2020) to generate candidates, and transform the original text span extraction into the candidate classification for tracking locations. Specifically, we first extract the POS tags by flair (Akbik et al. 2019), and then generate location candidates by POS-based rules[3].

For the train and dev sets, if the gold location is not included in the candidates, we manually add them to the candidate set. This is mainly for expanding the size of trainable instances in location prediction. For the test set, we do

[1] In procedural text understanding task, the division of timesteps is consistent with the division of sentences.

[2] *Other (O)* is further devided into O_A, O_B, which mean none state before and after existence separately.

[3] See details at https://github.com/ytyz1307zzh/NCET-ProPara.


[CLS] water [SEP] mineral [SEP] this combination of water and minerals ***flows*** from the stem into the leaf . [SEP]

S_3: this combination of water and minerals ***flows*** from the stem into the leaf.

water	E, soil	M, root	M, leaf	E, leaf	D, -	O_B, -
mineral s	E, soil	M, root	M, leaf	E, leaf	D, -	O_B, -

Figure 2: An overview of the proposed SGR in the scene-wise paradigm, which is composed of four parts: (a) graph structure encoder; (b) context encoder; (c) graph structure predictor and (d) state reasoner.

not use such method because we obviously cannot know the gold location while testing.

Scene Graph Reasoner

In this section, we describe how to train an effective procedural text understanding model in the scene-wise paradigm, and track state changes and locations of all entities simultaneously. As illustrated in Figure 2, we propose **S**cene **G**raph **R**easoner (**SGR**), a specific implementation for scene-wise procedural text understanding. SGR constructs scene graphs for each training instance. To evolve the scene graphs, SGR first summarizes critical information from the current scene graph by a graph structure encoder, captures the new events occurring from the sentence describing next narrative timestep by a context encoder, and then predicts the evolution of the scene graph after the new events by a graph structure predictor. During testing, SGR utlizes a state reasoner to simultaneously distills the state changes and locations of all entities via comparing the adjacent scene graphs.

Scene Graph Construction for Training

For each training instance, we transform the original gold state change and location annotations $\{Y^s, Y^l\}$ into a sequence of scene graphs Y^g to adapt for the proposed scene-wise paradigm, where $Y^g = \{y^g_1, y^g_2, ..., y^g_t, ..., y^g_T\}$. Each node in the scene graph represents a concept (entities, locations or elements from external knowledge), and each edge represents a predefined relation between two concepts. As the narrative develops, the nodes and edges in the scene graphs are dynamically created or deleted. Enlightened by (Skardinga, Gabrys, and Musial 2021), we utilize a complete graph with two matrices to record the dynamics of nodes and edges. And each scene graph can be represented as $y^g_t = \{\hat{\mathcal{G}}, Mask^t, Rel^t\}$: $\hat{\mathcal{G}}$ for the complete graph, $Mask^t$ for node masking and Rel^t for edge indicating at time t. Consequently, the model training objectives is reformulated to predict these scene graphs.

Specifically, SGR first uses the recognized entities and the generated location candidates as nodes[4], and use three SRL-based relations (*entity-entity, location-location and entity-location*) as edges to construct the complete graph. Then SGR enhances the complete graph with the external commonsense knowledge from ConceptNet (Speer, Chin, and Havasi 2017) because it can provide abundant concept relation, and help the model to understand composition of the world[5]. Finally SGR generates the node mask $Mask^t$ and the edge indication Rel^t for each scene graph y^g_t, where $Mask^t \in \mathbb{R}^M$ masks the entities that are not created or are already destroied, $Rel^t \in \mathbb{R}^{M*M*R}$ indicates different relations whose arguments are not masked, M is the number of concepts, and R is the number of relations.

In this way, the state changes and locations can be jointly modeled in the scene graphs, e.g., the state changes like *Exist*, *Create* and *Destroy* are record by $Mask^t$; the location of each entity are record by Rel^t.

[4] It is worth to notice that we do not treat events/actions as one kind of nodes as Huang et al. (2021) dose, because events are more suitable for evolving scene graphs.

[5] We retrieve and add the corresponding entities and relations (e.g, *HasA*, *PartOf*, and so on) into complete graphs using the same heuristic rules as Zhang et al. (2020).

Graph Structure Encoder for Summarizing Scenes

At timestep t, SGR adopts a graph attention network (GAT) (Velickovic et al. 2018) to summerize critical information from the current scene graph y_t^g since its strong representation capacity. In this way, the entities and their states/locations are jointly modeled by rich types of relations among different concepts.

Specifically, the input to the graph attention network is a set of node features $H = \{\hat{h}_1^{s,t}, \hat{h}_2^{s,t}, ..., \hat{h}_M^{s,t}\}$ at timestep t, where M is the number of concepts[6]. SGR then performs self-attention on the nodes — a shared masked attentional mechanism computes attention coefficients:

$$e_{ij}^t = a(\mathbf{W_1}\hat{h}_i^{s,t}, \mathbf{W_1}\hat{h}_j^{s,t}, \mathbf{W_2}Rel_{ij}^t) \qquad (1)$$

that indicate the importance of node j to node i, where Rel_{ij}^t is the relation embedding, $\mathbf{W_1}, \mathbf{W_2}$ are learnable parameters. To make coefficients easily comparable across different nodes, we normalize them across all choices of j using the softmax function:

$$\alpha_{ij}^t = softmax_j(e_{ij}^t) = \frac{exp(e_{ij}^t)}{\sum_{k \in \mathcal{N}_i^t} exp(e_{ik}^t)} \qquad (2)$$

where $\mathcal{N}_i^t$ is the neighborhood of node i in the scene graph at timestep t, which is determined by the complete graph $\hat{\mathcal{G}}$, the node mask $Mask^t$ and the edge indication Rel^t. Once obtained, the normalized attention coefficients are used to compute a linear combination of the features corresponding to them, to serve as the final features for every node at timestep t:

$$h_i^{s,t} = \sigma(\sum_{j \in \mathcal{N}_i^t} \alpha_{ij}^t \hat{h}_j^{s,t}) \qquad (3)$$

Finally, we obtain the hidden state corresponding to the special $[Global]$ node as the graph structure representation:

$$h_{[Gloabl]}^{s,t} = GAT(y_t^g) = GAT(\{\hat{\mathcal{G}}, Mask^t, Rel^t\}) \qquad (4)$$

where $h_{[Gloabl]}^{s,t}$ summerizes critical information from the current scene graph before new events occur. The scene graph structure and the enhanced external knowledge can be fully learned by the graph structure encoder.

Context Encoder for Capturing New Events

Existing procedural text understanding models do have the context encoder to obtain document-entity representations (Mishra et al. 2018; Du et al. 2019a,b; Tang, Feng, and Zhao 2020; Gupta and Durrett 2019b). However, we leverage the power of context encoder differently. We utilize it to capture the new events occurring in the sentence at the next timestep $t + 1$. In this paper, we use BERT (Devlin et al. 2019) to handle the nuances of procedural texts. As suggested by Gupta and Durrett (2019a), we restructure the input to guide the transformer model (Vaswani et al. 2017) to focus on particular entities mentioned in the sentence.

Specifically, take S_3 in Figure 2 as an example, we first restructure the input as: {*[CLS] water [SEP] minerals [SEP] This combination of water and minerals flows from the stem into the leaf . [SEP]*}, where *[CLS]* and *[SEP]* are special tokens. In this way, the transformer can always observe the entities it should be primarily "attending to" from the standpoint of building representations. For each token in the input, its representation is constructed by concatenating the corresponding token and position embeddings. Then, the context representation will be inputted into BERT architecture (Devlin et al. 2019), and updated by multilayer Transformer blocks (Vaswani et al. 2017).

Finally, we obtain the hidden state corresponding to the special *[CLS]* token in the last layer as the context representation:

$$h_{[CLS]}^{c,t+1} = BERT(restructure(S_{t+1})) \qquad (5)$$

where $h_{[CLS]}^{c,t+1}$ captures the new events occured in the sentence S_{t+1} at the current timestep $t + 1$. The self-attention mechanism of BERT supports the interactions of the multiple relationships between entities, locations and events mentioned in the sentence S_{t+1}. And it can take advantage of the knowledge learned via pretraining.

Graph Structure Predictor for Evolving Scenes

Based on the the structure representation $h_{[Gloabl]}^{s,t}$ and the context representation $h_{[CLS]}^{c,t+1}$, we can predict the new scene graph structure at timestep $t + 1$.

Specifically, we generate the new scene graph y_{t+1}^g via predicting the node mask $Mask^{t+1}$ and the edge indication Rel^{t+1} with the guidance of the aggregate representation:

$$\begin{aligned} \hat{Mask}^{t+1} &= f_1(h_{[Gloabl]}^{s,t}, h_{[CLS]}^{c,t+1}) \\ \hat{Rel}^{t+1} &= f_2(h_{[Gloabl]}^{s,t}, h_{[CLS]}^{c,t+1}) \end{aligned} \qquad (6)$$

where $\hat{Mask}^{t+1} \in \mathbb{R}^M$, $\hat{Rel}^{t+1} \in \mathbb{R}^{M*M*R}$, f_1, f_2 are two nonlinear output layers. And a sequence of scene graphs can be generated through an autoregressive behavior:

$$y_{t+1}^g = SGR(y_t^g, S_{t+1}) \qquad (7)$$

Model Training

Given a training corpus with constructed scene graphs (see Section **Scene Graph Construction for Training.**), SGR can be supervisedly learned by maximum log-likelihood estimation (MLE):

$$\begin{aligned} \mathcal{L} = &\sum_{t=1}^{T}\sum_{i=1}^{M} Mask_i^t \log \hat{Mask}_i^t \\ &+ \sum_{t=1}^{T}\sum_{i=1}^{M}\sum_{j=1}^{M}\sum_{k=1}^{R} Rel_{ijk}^t \log \hat{Rel}_{ijk}^t \end{aligned} \qquad (8)$$

where T is the number of timesteps, M is the number of concepts, including entities, location candidates and elements from external knowledge and R is the number of relations.

[6] At timestep 0, node features are initialized by context encoder.

Algorithm 1: : State Reasoner.

```
Input: SGR: the trained procedural text understanding model;
    ConceptNet: the external commonsense knowledge base;
    $P = \{S_1, S_2, ..., S_T\}$: the procedural text;
    $E = \{e_1, e_2, ..., e_N\}$: pre-specified entities;
    $Constraints$: used for postprocessing;
 1: The complete graph $\mathcal{G} \leftarrow$ Construct$(P, E)$
 2: The enhanced complete graph $\hat{\mathcal{G}} \leftarrow$ Enhance($\mathcal{G}$, ConceptNet)
 3: $Y^g \leftarrow \emptyset$
 4: $y^g_0 \leftarrow (\hat{\mathcal{G}}, Mask^0, Rel^0) \leftarrow (\hat{\mathcal{G}}, \emptyset, \emptyset)$
 5: $Y^g$.append$(y^g_0)$
 6: for $S_t$ in $P$ do
 7:     $h^{s,t}_{[Gloabl]} \leftarrow$ Graph Structure Encoder$(y^g_t)$
 8:     $h^{c,t+1}_{[CLS]} \leftarrow$ Context Encoder$(S_{t+1})$
 9:     $y^g_{t+1} \leftarrow (\hat{\mathcal{G}}, \hat{Mask}^t, \hat{Rel}^t) \leftarrow$
            Graph Structure Predictor$(h^{s,t}_{[Gloabl]}, h^{c,t+1}_{[CLS]})$
10:     $Y^g$.append$(y^g_{t+1})$
11: $(Y^s, Y^l) \leftarrow$ Transform$(\hat{\mathcal{G}}, Y^g, Constraints)$
Return: $Y^s, Y^l$;
```

State Reasoner for Tracking All Entities

During testing, on the basis of the trained procedural text understanding model SGR, we can construct scene graphs for new narratives, and then simultaneously track the state changes and locations of all entities scene-by-scene. Specifically, we can infer the state change and location sequences via comparing the adjacent scene graphs, e.g., if we find out that "*water*" is not masked at scene y^g_{t-1} but masked at scene y^g_t, we can deterministically infer the state of "*water*" is "*Destroy (D)*" at timestep t; if we find that "*water*" has a "*LocateIn*" realtion with "*root*", the current location of "*water*" must be "*root*". It is worth to notice that these transformations of all entities can be processed in parallel.

To facilitate the description of state reasoner, we summarize this process in Algorithm 1. Specifically, we first preprocess the raw input $\{P, E\}$ to construct the complete graph, and then utilize the external commonsense knowledge ConceptNet (Speer, Chin, and Havasi 2017) to get the enriched complete graph (**Line 1-2**). Second, we initialize the scene graphs Y^g as $\emptyset$ (**Line 3**). However, during testing, we cannot access the gold state change and location annotations. Thus, we initialize $Mask^0$ and Rel^0 as zeros, which means that we know nothing about the current world at timestep 0 (**Line 4**). After these preparations, we utilize the trained model SGR to evolve the scene graphs from y^g_0 to y^g_T, which contains graph structure encoding, context encoding and graph structure predicting (**Line 6-10**). Finally, we transform the scene graphs Y^g into the state change sequence Y^s and the location sequence Y^l for all pre-specified entities (**Line 6-11**). The constraints used in previous works can be easily inject into the final transformation process, e.g., correct invalid actions according to the whole action sequence (Tang, Feng, and Zhao 2020).

In this way, the state change and location sequences of all entities can be tracked simultaneously, and the efficiency of reasoning can be significantly improved.

Statistics	**ProPara**			**Recipes**		
	train	**dev**	**test**	**train**	**dev**	**test**
#Instance/Para	391	43	54	693	86	87
#Sentence	2,620	288	372	6,098	765	783
#Entity	1,504	175	236	5,932	756	737
Avg.#sent/para	6.7	6.7	6.9	8.8	8.9	9.0
Avg.#enti/para	3.8	4.1	4.4	8.6	8.8	8.5

Table 1: Statistics of ProPara and Recipes datasets. We regard one paragraph with all pre-specified entities as an instance. Thus, the number of instances is equivalent to the number of paragraphs.

Experiments

Experimental Settings

Dataset. We conduct main experiments on ProPara (Mishra et al. 2018) and auxiliary experiments on Recipes (Bosselut et al. 2018). For ProPara, we follow the official split (Mishra et al. 2018) for train/dev/test set. For Recipes, following the previous works (Zhang et al. 2020; Huang et al. 2021), we only use the human-labeled data in our experiments, and re-split it into 80%/10%/10% for train/dev/test sets. More statistics about these two datasets are shown in Table 1[7].

Implementation Details. For graph structure encoder, we apply a one-layer graph attention network (GAT) (Velickovic et al. 2018). For context encoder, we use the BERT base implemented by HuggingFace's transformers library (Wolf et al. 2020). Hyper-parameters are manually tuned according to the accuracy on the dev set: batch size is set to 16, hidden size is set to 128 and learning rate is set to 5e-5. The final model is trained on an Nvidia TITAN RTX GPU with Adam optimizer (Kingma and Ba 2015), and is selected with the highest prediction accuracy on dev set.

Evaluation Metrics For ProPara, following the previous works, we perform document level (Tandon et al. 2018) and sentence level (Mishra et al. 2018) tasks in our main experiments. Specifically, the document level task requires models to answer the four document-level questions:

Q1: What are the inputs to the procedure?

Q2: What are the outputs of the procedure?

Q3: What conversions occur, when and where?

Q4: What movements occur, when and where?

The evaluator compute precision, recall and F1 score for each question, and the overall F1 score is the macro-average of the above four questions[8]. The sentence-level task requires models to answer ten fine grained sentence-level questions, which can be summarized into three categories:

Cat-1: Is entity created (destroyed, moved)?

Cat-2: When is entity created (destroyed, moved)?

Cat-3: Where is entity created (destroyed, moved from/to)?

[7] The number of instances is different from the previous entity-wise works because they regard one entity-paragraph pair as an instance, and result in 1.9k instances

[8] https://github.com/allenai/aristo-leaderboard/tree/master/ProPara

Models	Document-level task			Sentence-level task				
	Precision	Recall	F1	Cat-1	Cat-2	Cat-3	Macro-Avg	Micro-Avg
Entity-wise Models								
Models with context encoder								
EntNet (Henaff et al. 2017)	54.7	30.7	39.4	51.6	18.8	7.8	26.1	26.0
QRN (Seo et al. 2017)	60.9	31.1	41.4	52.4	15.5	10.9	26.3	26.5
ProLocal (Mishra et al. 2018)	81.7	36.8	50.7	62.7	30.5	10.4	34.5	34.0
ProGlobal (Mishra et al. 2018)	48.8	61.7	51.9	63.0	36.4	35.9	45.1	45.4
• AQA (Ribeiro et al. 2019)	62.0	45.1	52.3	61.6	40.1	18.6	39.4	40.1
• ProStruct (Tandon et al. 2018)	74.3	43.0	54.5	-	-	-	-	-
XPAD (Du et al. 2019a)	70.5	45.3	55.2	-	-	-	-	-
LACE (Du et al. 2019b)	75.3	45.4	56.6	-	-	-	-	-
NCET (Gupta and Durrett 2019b)	67.1	58.5	62.5	73.7	47.1	41.0	53.9	54.0
◇ ET_BERT (Gupta and Durrett 2019a)	-	-	-	73.6	52.6	-	-	-
∗ IEN (Tang, Feng, and Zhao 2020)	69.8	56.3	62.3	71.8	47.6	40.5	53.3	53.0
◇ DYNAPRO (Amini et al. 2020)	75.2	58.0	65.5	72.4	49.3	**44.5**	55.4	55.5
• ◇ KOALA (Zhang et al. 2020)	77.7	**64.4**	70.4	78.5	53.3	41.3	57.7	57.5
Models with structure encoder								
KG-MRC (Das et al. 2019)	69.3	49.3	57.6	62.9	40.0	38.2	47.0	46.6
• ProGraph (Zhong et al. 2020)	67.3	55.8	61.0	67.8	44.6	41.8	51.4	51.5
◇ TSLM (Faghihi and Kordjamshidi 2021)	68.4	68.9	68.6	78.8	56.8	40.9	58.8	58.3
◇ REAL (Huang et al. 2021)	81.9	61.9	70.5	78.4	53.7	42.4	58.2	57.9
Scene-wise Models								
• ◇ ∗ SGR (our method)	**84.9**	62.9	**72.2**	**79.9**	55.1	43.5	**59.5**	**59.2**
w/o Graph Structure Encoder	72.4	51.1	59.9	69.9	42.7	39.9	50.8	51.0
w/o Context Encoder	76.1	55.4	64.1	74.9	47.9	40.0	54.3	54.2
w/o ConceptNet	82.7	63.2	71.6	78.3	**56.0**	42.5	58.9	58.6
w/o Pre-trained Bert	81.8	59.2	68.7	76.2	53.3	41.4	57.0	56.7

Table 2: Experimental results on ProPara document-and sentence-level tasks. ∗, • and ◇ indicate the models consider the interactions between multiple entities, use the external knowledge base and are equipped with the pre-trained model separately.

Models	Precision	Recall	F1
NCET (re-implementation)	56.5	46.4	50.9
IEN (re-implementation)	58.5	47.0	52.2
KOALA (Zhang et al. 2020)	60.1	52.6	56.1
REAL (Huang et al. 2021)	55.2	**52.9**	54.1
SGR (our method)	**69.3**	50.5	**58.4**

Table 3: Experimental results on re-split Recipes.

Evaluation metrics are macro-average and micro-average accuracy of three sets of questions. More details can be found in the official script[9]. The answers of both document- and sentence-level questions can be deterministically computed from the state change and location sequences.

For Recipes, we follow Zhang et al. (2020); Huang et al. (2021) to predict the location changes of the ingredients during the procedure. For each movement, the model should predict the new location of the entity, plus the timestep when the movement occurs. We take precision, recall, and F1 scores to evaluate models.

Baselines

For ProPara, we compare SGR with the following baselines, most of them are on the official leaderboard[10]:

- **Models with context encoder** rely on Bi-LSTM/Bert to obtain the document-entity representations and track the states/locations separately.
- **Models with structure encoder** leverage the power of the static graph to obtaion more effective document representations. Different from our work, they lack the dynamical representations of the procedures.

For Recipes, we compare SGR with the state-of-the-art models: NCET (Tang, Feng, and Zhao 2020), KOALA (Zhang et al. 2020) and REAL (Huang et al. 2021).

Overall Results

Table 2, 3 and 4 show the overall results. We can see that:

1. The proposed SGR in the scene-wise paradigm achieves the state-of-the-art performance. SGR can significantly outperform the state-of-the-art model REAL and achieves 72.2 F1 on ProPara document-level task and 59.5/59.2 Macro-Avg/Micro-Avg scores on ProPara sentence-level task. On Recipes, SGR also outperforms the corresponding baselines and achieves 58.4 F1. We believe this is because that the entities and their states/locations are jointly modeled in the scenes. Therefore, the association of two track targets and the interaction of the multiple entities are fully explored.

2. Reasoning states and locations of all entities scene-by-scene significantly improves the inference efficiency. We compare the inferencing time of SGR with NCET and IEN on an Nvidia TITAN RTX GPU in Table 4. NCET

[9] https://github.com/allenai/ProPara/tree/master/ProPara/evaluation

[10] https://leaderboard.allenai.org/ProPara/submissions/public

Models	ProPara		
	Total	Avg./para	Avg./enti
NCET (re-implementation)	51.31	0.95	0.22
IEN (re-implementation)	42.50	0.79	0.18
SGR (our method)	**17.99**	**0.33**	**0.08**

Table 4: Inference time (seconds) on ProPara.

	ConceptNet		Document-level task		
	Train	Test	Precision	Recall	F1
SGR △	✓	✓	**84.9**	62.9	**72.2**
SGR(test) △		✓	83.7	62.9	71.8
SGR(train) ▽	✓		81.5	60.4	69.4
SGR(none)			82.7	**63.2**	71.6

Table 5: Effect of the usage of ConceptNet on ProPara. All improvements of SGR are statistical significance at $p<0.01$.

is the traditional model in the entity-wise paradigm, while IEN considers the interactions between multiple entities via the entity-location attention mechanism[11]. However, both NCET and IEN rely on separate trackers with CRF to predict state changes and locations. Thus they are time-intensive and take almost 3-4 times as long as SGR to track each entity. These results verifies that reasoning questions scene-by-scene is efficient and promotes the application of the procedural text understanding model in real-world scenes.

3. The graph structure encoder and the context encoder are indispensable, and are complementary with each other. When compared with the full model SGR, its two variants SGR w/o Graph Structure Encoder and SGR w/o Context Encoder show declined performance in different degrees, which indicates that the current scene modeled by the graph structure encoder and the new events captured by the context encoder are necessary. Surprisingly, we find that SGR w/o Context Encoder still perform quite well. The insight in those observations may be that it can be regard as a graph-structured language model — predicts snapshots through an autoregressive behavior, and builds label consistency in the same topic (Du et al. 2019b).

Detailed Analysis

The external knowledge can be easily injected into SGR. To investigate the effectiveness of difference kinds of external knowledge, we design the following experiments.

Effects of the External Knowledge from the Pre-trained language model. From Table 2, we can see that models indicates with ⋄ outperform other baselines. When compared with SGR, the performance of SGR w/o Pre-trained BERT clampes between SGR and SGR w/o Context Encoder. It means that not only the context encoder but also the external knowledge learned via pre-training is helpful for procedural text understanding.

Effects of the External Knowledge from the Knowledge Base. First of all, from Table 2, we can see that models indicates with • outperform other baselines. And the decay of SGR w/o ConceptNet is also appreciable when compared with SGR. These results verify the effectiveness of the external knowledge from knowledge base. Furthermore, we investigate the usage of ConceptNet in Table 5. We can see that: 1) Compared with SGR(none), SGR and SGR(test) lead to improvements. The reason behinds it is that ConceptNet can constrain the prediction space of graph structure predictor and help the model to understand composition of the world. 2) SGR(train) even perform worse than SGR(none). It is because that the inconsistency between train and test introduces too many noisies rather than knowledge. In other words, the exposure bias of the autoregressive behavior hurts the performance of the model (Zhang et al. 2019).

Related Work

Procedural Text Understanding is important and challenging. Many datasets have been proposed such as bAbI (Henaff et al. 2017), RECIPES (Kiddon et al. 2015) and ProPara (Mishra et al. 2018). Blessed with valuable benchmarks, there emerge abundant procedural text understanding models which are in the question-answering framework (Henaff et al. 2017; Seo et al. 2017; Das et al. 2019) or hierarchical neural network framework (Mishra et al. 2018; Tandon et al. 2018; Du et al. 2019b; Gupta and Durrett 2019b,a; Zhang et al. 2020; Zhong et al. 2020). Some of them utilize graph encoder to obtaion more effective document-entity representations (Huang et al. 2021) and almost of them in the entity-wise paradigm. Different from them, this paper propose a new scene-wise paradigm to jointly tracks the state changes and locations of all entities scene-by-scene.

Dynamic Graph Neural Networks (DGNNs) are used in a wide range of fields, including social network analysis, recommender systems and epidemiology (Yin et al. 2019; Skardinga, Gabrys, and Musial 2021). DGNNs add a new dimension to network modeling and prediction – time. This new dimension radically influences network properties which enable a more powerful representation of network, and increases predictive capabilities of methods (Aggarwal and Subbian 2014; Li et al. 2018). In this paper, we utilize a graph structure to model scene, and an evaluation algorithm is proposed to adapt for the proposed scene-wise paradigm.

Conclusions

In this paper, we propose a new **scene-wise** paradigm for procedural text understanding and **S**cene **G**raph **R**easoner (**SGR**) is designed to jointly model the associations of state changes and locations, as well as the interactions of multiple entities. In this way, the state changes and locations of all entities are jointly exploited and then can be simultaneously derived from the scene graphs. Experiments show that SGR achieves the new state-of-the-art procedural text understanding performance, and the reasoning speed is significantly accelerated. For future work, we want to pretrain a graph-structured language model to build label consistency in the same topic (Du et al. 2019b) and design new training and reasoning methods to overcome the exposure bias of the autoregressive behavior (Zhang et al. 2019).

[11] Other state-of-the-art models spend more reasoning times than NCET and IEN due to exquisitely designed archectures.

Acknowledgments

We sincerely thank the reviewers for their insightful comments and valuable suggestions. This research work is supported by the National Key R&D Program of China (2020AAA0105200), and the National Natural Science Foundation of China under Grants no. U1936207, 62122077, 62106251 and 61772505, the Beijing Academy of Artificial Intelligence (BAAI) and in part by the Youth Innovation PromotionAssociation CAS(2018141).

References

Aggarwal, C.; and Subbian, K. 2014. Evolutionary Network Snalysis: A Survey. *ACM Computing Surveys (CSUR)*.

Akbik, A.; Bergmann, T.; Blythe, D.; Rasul, K.; Schweter, S.; and Vollgraf, R. 2019. FLAIR: An Easy-to-use Framework for State-of-the-art NLP. In *Proceedings of NAACL 2019*.

Amini, A.; Bosselut, A.; Mishra, B. D.; Choi, Y.; and Hajishirzi, H. 2020. Procedural Reading Comprehension with Attribute-Aware Context Flow. In *Proceedings of AKBC 2020*.

Bosselut, A.; Levy, O.; Holtzman, A.; Ennis, C.; Fox, D.; and Choi, Y. 2018. Simulating Action Dynamics with Neural Process Networks. In *Proceedings of ICLR 2018*.

Clark, C.; and Gardner, M. 2018. Simple and Effective Multi-Paragraph Reading Comprehension. In *Proceedings of ACL 2018*.

Das, R.; Munkhdalai, T.; Yuan, X.; Trischler, A.; and McCallum, A. 2019. Building Dynamic Knowledge Graphs from Text using Machine Reading Comprehension. In *Proceedings of ICLR 2019*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL 2019*.

Du, X.; Mishra, B. D.; Tandon, N.; Bosselut, A.; tau Yih, W.; and Clark, P. 2019a. Everything Happens for a Reason: Discovering the Purpose of Actions in Procedural Text. In *Proceedings of EMNLP 2019*.

Du, X.; Mishra, B. D.; Tandon, N.; Bosselut, A.; tau Yih, W.; Clark, P.; and Cardie, C. 2019b. Be Consistent! Improving Procedural Text Comprehension using Label Consistency. In *Proceedings of NAACL 2019*.

Faghihi, H. R.; and Kordjamshidi, P. 2021. Time-Stamped Language Model: Teaching Language Models to Understand The Flow of Events. In *Proceedings of NAACL 2021*.

Gupta, A.; and Durrett, G. 2019a. Effective Use of Transformer Networks for Entity Tracking. In *Proceedings of EMNLP 2019*.

Gupta, A.; and Durrett, G. 2019b. Tracking Discrete and Continuous Entity State for Process Understanding. In *Proceedings of NAACL 2019 Workshop*.

Henaff, M.; Weston, J.; Szlam, A.; Bordes, A.; and LeCun, Y. 2017. Tracking the World State with Recurrent Entity Networks. In *Proceedings of ICLR 2017*.

Huang, H.; Geng, X.; Pei, J.; Long, G.; and Jiang, D. 2021. Reasoning over Entity-Action-Location Graph for Procedural Text Understanding. In *Proceedings of ACL 2021*.

Kiddon, C.; Ponnuraj, G. T.; Zettlemoyer, L.; and Choi, Y. 2015. Mise En Place: Unsupervised Interpretation of Instructional Recipes. In *Proceedings of EMNLP 2015*.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of ICLR 2015*.

Li, T.; Wang, B.; Jiang, Y.; Zhang, Y.; and Yan, Y. 2018. Restricted Boltzmann Machine-based Approaches for Link Prediction in Dynamic Networks. *IEEE Access*.

Mishra, B. D.; Huang, L.; Tandon, N.; tau Yih, W.; and Clark, P. 2018. Tracking State Changes in Procedural Text: A Challenge Dataset and Models for Process Paragraph Comprehension. In *Proceedings of NAACL 2018*.

Ribeiro, D.; Hinrichs, T.; Crouse, M.; Forbus, K.; Chang, M.; and Witbrock, M. 2019. Predicting State Changes in Procedural Text using Analogical Question Answering. In *Proceedings of ACACS 2019*.

Seo, M. J.; Kembhavi, A.; Farhadi, A.; and Hajishirzi, H. 2017. Bidirectional Attention Flow for Machine Comprehension. In *Proceedings of ICLR 2017*.

Skardinga, J.; Gabrys, B.; and Musial, K. 2021. Foundations and Modelling of Dynamic Networks using Dynamic Graph Neural Networks: A survey. *IEEE Access*.

Speer, R.; Chin, J.; and Havasi, C. 2017. Conceptnet 5.5: An Open Multilingual Graph of General Knowledge. In *Proceedings of AAAI 2017*.

Tandon, N.; Mishra, B. D.; Grus, J.; tau Yih, W.; Bosselut, A.; and Clark, P. 2018. Reasoning about Actions and State Changes by Injecting Commonsense Knowledge. In *Proceedings of EMNLP 2018*.

Tang, J.; Feng, Y.; and Zhao, D. 2020. Understanding Procedural Text using Interactive Entity Networks. In *Proceedings of EMNLP 2020*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Kaiser, A. N. G. Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Proceedings of NIPS 2017*.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2018. Graph Attention Networks. In *Proceedings of ICLR 2018*.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; Davison, J.; Shleifer, S.; von Platen, P.; Ma, C.; Jernite, Y.; Plu, J.; Xu, C.; Scao, T. L.; Gugger, S.; Drame, M.; Lhoest, Q.; and Rush, A. M. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of EMNLP 2020: System Demonstrations*.

Yin, Y.; Song, L.; Su, J.; Zeng, J.; Zhou, C.; and Luo, J. 2019. Graph-based Neural Sentence Ordering. In *Proceedings of IJCAI 2019*.

Zhang, W.; Feng, Y.; Meng, F.; You, D.; and Liu, Q. 2019. Bridging the gap between training and inference for neural machine translation. In *Proceedings of ACL 2019*.

Zhang, Z.; Geng, X.; Qin, T.; Wu, Y.; and Jiang, D. 2020. Knowledge-aware Procedural Text Understanding with Multi-stage Training. In *Proceedings of WWW 2020*.

Zhong, W.; Tang, D.; Duan, N.; Zhou, M.; Wang, J.; and Yin, J. 2020. A Heterogeneous Graph with Factual, Temporal and Logical Knowledge for Question Answering Over Dynamic Contexts. *ArXiv preprint arXiv:2004.12057*.