Understanding Procedural Text using Interactive Entity Networks

Jizhi Tang, Yansong Feng, Dongyan Zhao
Wangxuan Institute of Computer Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
{tangjizhi, fengyansong, zhaody}@pku.edu.cn

Abstract

The task of procedural text comprehension aims to understand the dynamic nature of entities/objects in a process. Here, the key is to track how the entities interact with each other and how their states are changing along the procedure. Recent efforts have made great progress to track multiple entities in a procedural text, but usually treat each entity separately and ignore the fact that there are often multiple entities interacting with each other during one process, some of which are even explicitly mentioned. In this paper, we propose a novel Interactive Entity Network (IEN), which is a recurrent network with memory equipped cells for state tracking. In each IEN cell, we maintain different attention matrices through specific memories to model different types of entity interactions. Importantly, we can update these memories in a sequential manner so as to explore the causal relationship between entity actions and subsequent state changes. We evaluate our model on a benchmark dataset, and the results show that IEN outperforms state-of-the-art models by precisely capturing the interactions of multiple entities and explicitly leverage the relationship between entity interactions and subsequent state changes. Our code is available at: https://github.com/esddse/IEN.

1 Introduction

Procedural texts, e.g., scientific articles, instruction books, or recipes, are widely spread and useful. Entity state tracking is the key task for procedural text comprehension. Usually, an entity could have several targets (e.g., existence, location) to be tracked, and the system needs to predict the target changes of each entity involved in the process. State tracking is challenging because of the dynamic nature, the involvement of multiple entities, and the complexity of tracking targets.

Most recent approaches often use an RNN-based method to model the state changes across the process in an entity-by-entity manner, and use different classifiers for different targets (Dalvi et al., 2018; Tandon et al., 2018; Gupta and Durrett, 2019).

While these models can learn to leverage either local or global information and make state predictions to one entity with fair accuracy, they ignore the fact that there are often multiple entities interacting with each other during the procedure. Here is an example that describes a chemical transformation step in the photosynthesis process:

> The water breaks into oxygen, hydrogen, and electrons.

To perfectly capture this chemical conversion in the process, "entity-by-entity" systems need to separately find out that water is destroyed at this step, oxygen is created, hydrogen is created, and electrons are created at the same step, respectively. It is easy to see that if the system ignores the interactions among multiple entities, it will be prone to misunderstanding the role of different entities in the conversion, and consequently make similar state prediction for water as the other entities, although they are opposite in fact.

Another challenge is how to properly capture the relationship between entity interactions and their state changes. For example, consider the following steps in blood circulation:

> Blood travels to the lungs.
> Carbon dioxide is removed from the blood.
> Oxygen is added to your blood.

In order to figure out where the carbon dioxide is after the second sentence, humans may first want to see what role the carbon dioxide acts in that sentence, and quickly find that carbon dioxide is just taken away from the blood thus impossible to be in the blood. Furthermore if we look back to the first sentence, we can finally get the right answer, lung. This tells us that an entity's location is closely related or even determined by the most recent event it involves in. However, current systems either model a general state of an entity regardless of specific tracking targets (Dalvi et al., 2018), or model different tracking targets (e.g., action, location) completely separately (Gupta and Durrett, 2019), without considering the relationship between different tracking targets.

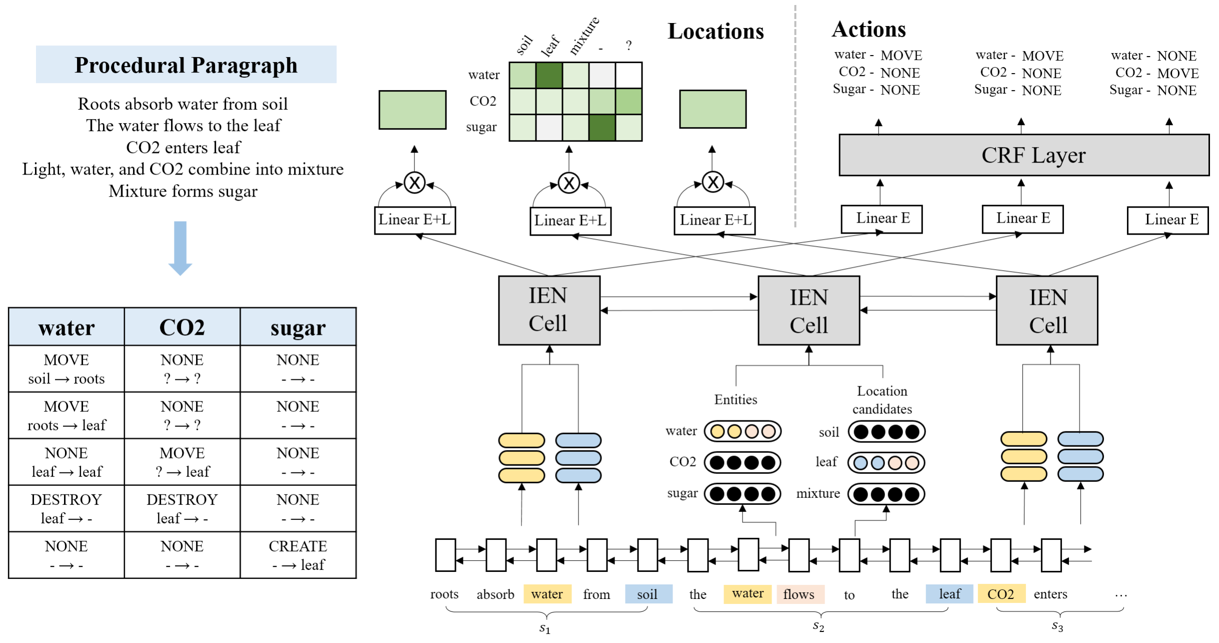
In this work, we focus on the scientific process understanding task (Dalvi et al., 2018), in which the tracking targets are action and location of entities. We propose a novel Interactive Entity Network, IEN, that explicitly models the interactions among multiple entities and explores the relationship between an entity's action and its location. IEN is a two-layer RNN model, the bottom RNN encodes word-level information, and the upper RNN encodes the sentence-level information while keeping tracking entities' states. Specifically, we design a new RNN cell, the IEN cell, for the upper RNN, which contains two types of memory slots, entity slots, and location slots, for recording the states of all entities and location candidates, respectively. To track the state changes, all memory slots will be updated at each time step. We use two attention modules to model entity-entity interaction and location-entity interaction, respectively, thus each entity/location slot is able to obtain information from other entities before updating. To model the causal relationship between different tracking targets, we arrange the entity updating module to precede the location updating module in a sequential manner. In this way, we can use the action-aware entity information to update the location slots. We evaluate IEN on ProPara, and the results show that our model can effectively model the interaction among multiple entities and leverage the causal relationship between entity actions and locations, thus outperforms existing systems.

Our contributions are in two-fold: 1) We propose a new model, IEN, and design a new recurrent unit that explicitly models the interactions among multiple entities and leverages the causal relationship between entity actions and their subsequent state changes. 2) We conduct intensive experiments to show how our IEN learns to encourage the synergy among different entities involved in one event, and explain how multiple tracking targets can be properly leveraged to improve context reasoning.

2 Related Work

Recently, many procedural text comprehension datasets are constructed and relesed to prompt the research in this direction. **bAbI** (Weston et al., 2015) is a QA dataset that the questions are about movements of entities, however it is synthetically generated and the language expression is relatively simple. **RECIPES** (Kiddon et al., 2015) dataset introduces the task of predicting the locations of cooking ingredients. **ProPara** (Dalvi et al., 2018) includes scientific procedural paragraphs, and the task is to predict the entities' actions and locations. In this paper, we continue this line of exploration using ProPara.

The solutions are mainly RNN based or memory network based. Most early models are designed for QA task, e.g., bAbI, and thus researchers pay more attention to question processing. **EntNet** (Henaff et al., 2016) uses dynamic memories to maintain entity states, with a gated update at each step. These states are decoded to answer questions after each sentence is read. **QRN** (Seo et al., 2016) is an RNN-based model. Given a question, QRN recurrently reduces the original query to a more informed query as it observes each context sentence through time. More recently, ProPara becomes the popular testbed and methods on ProPara focus more on state tracking. **ProLocal** (Dalvi et al., 2018) locally predicts the state changes described in each individual sentence. **ProGlobal** (Dalvi et al., 2018) considers the entire paragraph while predicting states for an entity. As the whole context is incorporated, more state changes are captured, and result in a higher recall. However, this may lead to over-prediction. To address these problems, several models are proposed to incorporate different constraints. **ProStruct** (Tandon et al., 2018) reformulate the procedural text comprehension task as a structured prediction task, and incorporates a set of commonsense constraints for globally consistent predictions. **LACE** (Du et al., 2019) leverages label consistency among different paragraphs on the same topic during training. **NCET** (Gupta and Durrett, 2019) uses a neural CRF to explicitly capture the constrains. Other interesting attempts include **KG-MRC** (Das et al., 2018), which constructs dynamic bipartite graphs from the procedural text, and updates the graphs


Procedural Paragraph

Roots absorb water from soil
The water flows to the leaf
CO2 enters leaf
Light, water, and CO2 combine into mixture
Mixture forms sugar

water	CO2	sugar
MOVE soil → roots	NONE ? → ?	NONE - → -
MOVE roots → leaf	NONE ? → ?	NONE - → -
NONE leaf → leaf	MOVE ? → leaf	NONE - → -
DESTROY leaf → -	DESTROY leaf → -	NONE - → -
NONE - → -	NONE - → -	CREATE - → leaf

Figure 1: The left is an example of the procedural text comprehension task and ProPara dataset. The paragraph describes the process of photosynthesis in five sentences, and the system needs to track the actions and locations of the three entities listed below, water, CO2, and sugar. The right is the architecture of IEN.

recurrently at each step.

3 Task Formulation

In this paper, we focus on the task of tracking multiple entities' states in scientific procedural text. Formally, given:

- A paragraph of procedural text $S = \{s_t\}_{t=1}^{T}$, consisting of T sentences, which describes a process about a given topic (e.g., photosynthesis, fossil formation) in detailed steps, with multiple entities involved.
- A set of entities $E = \{e_i\}_{i=1}^{n}$, representing the entities mentioned in S that need to be tracked.
- A set of tracking targets $P = \{p_j\}_{j=1}^{k}$. Each p_j is a specific property of an entity, e.g., existence or location, and each can have limited or infinite numbers of values.

The state tracking task is to predict the states of each entity e_i after reading each sentence s_t, where an entity's state is a value of a property p_j. For example, after reading *CO2 enters leaf*, the system should predict that the existence of CO2 is true, and the location of CO2 is *leaf*.

We use the ProPara dataset (Dalvi et al., 2018) for experiments. ProPara contains 488 procedural texts, 391 for training, 43 for development and 54 for testing. Each paragraph describes a particular scientific process. There are in total 3302 sentences, and 6.77 sentences per paragraph. On average, each paragraph contains 3.92 entities. The dataset tracks two types of entity property, action and location. For action, the model needs to determine which of the following actions the entity performed in the sentence: (1) CREATE, the entity is created at this step. (2) DESTROY, the entity is destroyed at this step. (3) MOVE, the entity is moved from one location to another. (4) NONE, none of above. For location, the model needs to determine where the entity locates before and after one sentence. The location could be any arbitrary span in the paragraph and needs to be extracted. There are two special locations: "-" denotes non-existence and "?" denotes unknown location. Figure 1 gives an example of ProPara dataset.

4 Methods

Our complete approach consists of 3 stages: 1) Preprocessing: we use off-the-shelf NLP tools to prepare the neural network inputs, including entity recognition and location candidates generation. 2) Main Model: we use Interactive Entity Network to predict all entities' actions and locations sequentially. 3) Postprocessing: we use human-written rules to revise some of the predictions to keep com-

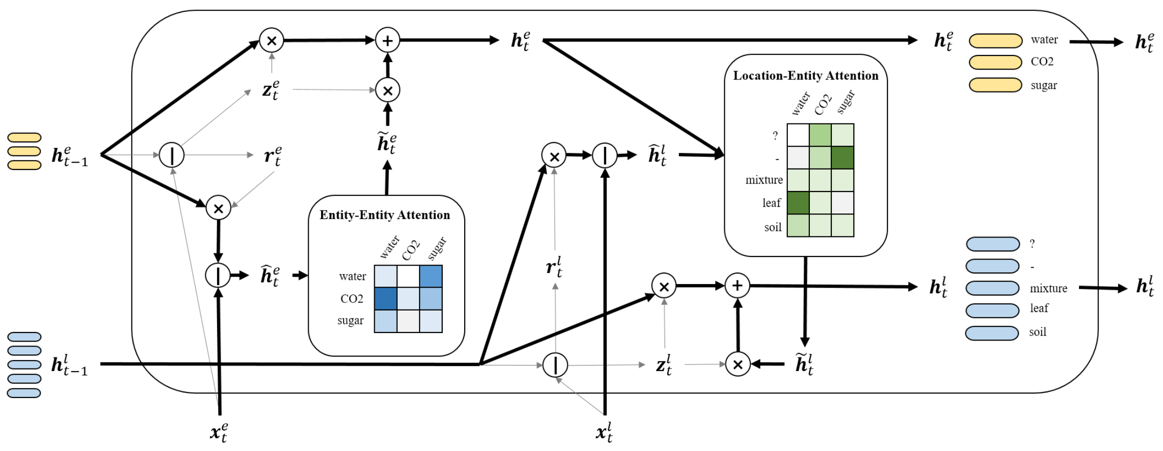


Figure 2: The structure of an IEN cell. The thick arrows indicate the directions of data flow.

monsense consistency.

4.1 Preprocessing

We use NLTK [1] and Stanford CoreNLP (Manning et al., 2014) for lemmatization and POS tagging. For entity recognition, we just use the simple string matching algorithm. Unlike entities, location information in this task is not given initially. To predict the entities' location, there are mainly two approaches. One is to view this problem as a SQuAD-style QA task(Rajpurkar et al., 2016), and to find a location span sentence by sentence from the paragraph, e.g., ProLocal (Dalvi et al., 2018) or KG-MRC (Das et al., 2018). The other approach is to first dig out all possible location candidates during preprocessing, and then use a classifier to select one candidate as the best location at each step, e.g., NCET (Gupta and Durrett, 2019). We use the latter paradigm and collect the location candidate set $L = \{l_j\}_{j=1}^{m}$ by gathering all nouns and noun phrases. This strategy has a 86.75% recall rate on the training set.

4.2 Interactive Entity Networks

We design a new model, Interactive Entity Network (IEN), to characterize the process with multiple entities involved. Figure 1 shows the architecture of our model. IEN consists of three levels. The bottom is for word-level language comprehension, which encodes words to distributed vectors. The middle is for sentence-level process understanding, which conducts entity state tracking. At the top, we use different classifiers to predict entities' actions and locations, respectively.

[1]http://www.nltk.org/

Word-Level Encoding Given a procedural text, our model first encodes each word w_i in the paragraph to a vector $\boldsymbol{w}_i = [\mathrm{emb}(w_i); v_i]$. Here, $\mathrm{emb}(w_i)$ is an embedding function, and we use fastText (Bojanowski et al., 2017) and ELMo (Peters et al., 2018) for experiments. v_i is a scalar binary indicator for identifying whether w_i is a verb. Then we use a BiLSTM (Hochreiter and Schmidhuber, 1997) over the whole paragraph for contextual encoding. We denote $\boldsymbol{u}_i = \mathrm{BiLSTM}([\boldsymbol{w}_i])$ as the output of BiLSTM with respect to word w_i.

Sentence-Level Encoding and IEN cell To track the state changes, we extract sentence features from word-level encodings by running another RNN at the sentence level. In order to take entity interactions into consideration, we propose a novel IEN cell that leverages attention mechanisms to help entities or locations get information from each other.

The inputs to an IEN cell include the representations of all entities and all location candidates in a single sentence s_t, or a mask vector if the entity or location candidate is not in s_t. Formally, let $\boldsymbol{x}_t^{e_i} \in \mathbb{R}^d$ denote the representation of entity e_i in sentence s_t, $\boldsymbol{x}_t^{l_j} \in \mathbb{R}^d$ denote the representation of location candidate l_j in sentence s_t. Then,

$$\boldsymbol{x}_t^{e_i} = \begin{cases} [\boldsymbol{u}_t^{e_i}; \boldsymbol{u}_t^{v}], & \text{if} \quad e_i \in s_t \\ \boldsymbol{0}, & \quad \text{otherwise} \end{cases} \tag{1}$$

$$\boldsymbol{x}_t^{l_j} = \begin{cases} [\boldsymbol{u}_t^{l_j}; \boldsymbol{u}_t^{v}], & \text{if} \quad l_j \in s_t \\ \boldsymbol{0}, & \quad \text{otherwise} \end{cases} \tag{2}$$

where $\boldsymbol{u}_t^e$, $\boldsymbol{u}_t^l$ and $\boldsymbol{u}_t^v$ denote the contextual encodings of the entity, location candidate and the predicate verb, respectively. If the entity or location candidate consists of multiple words, mean pooling over the word representations is used. We stack all the entity representations together to get $\boldsymbol{x}_t^e \in \mathbb{R}^{n\times d}$, and similarly get $\boldsymbol{x}_t^l \in \mathbb{R}^{m\times d}$. $\boldsymbol{x}_t^e$ and $\boldsymbol{x}_t^l$ are the inputs to the tth IEN cell.

Inspired by GRU (Chung et al., 2014) and key-value memory networks (Miller et al., 2016), we place memory slots inside IEN cells and let them recurrently update as GRU. In each IEN cell, there are n entity slots and m location slots, corresponding to the given entity set E and the extracted location candidate set L, respectively. Each memory slot represents the state of a specific entity or a location candidate. We use $\boldsymbol{h}_t^e \in \mathbb{R}^{n\times d}$ to represent all the entity memory slots in the tth IEN cell, and use $\boldsymbol{h}_t^l \in \mathbb{R}^{m\times d}$ to represent all the location memory slots in the tth IEN cell. The detailed structure of an IEN cell is shown in Figure 2.

First, we update the entity memory slots as follows:

$$\boldsymbol{z}_t^e = \sigma([\boldsymbol{h}_{t-1}^e; \boldsymbol{x}_t^e] \cdot \boldsymbol{W}_z^e) \tag{3}$$

$$\boldsymbol{r}_t^e = \sigma([\boldsymbol{h}_{t-1}^e; \boldsymbol{x}_t^e] \cdot \boldsymbol{W}_r^e) \tag{4}$$

$$\hat{\boldsymbol{h}}_t^e = [\boldsymbol{r}_t^e \odot \boldsymbol{h}_{t-1}^e; \boldsymbol{x}_t^e] \tag{5}$$

$$\tilde{\boldsymbol{h}}_t^e = \tanh(\mathrm{Att}(\hat{\boldsymbol{h}}_t^e, \hat{\boldsymbol{h}}_t^e, \hat{\boldsymbol{h}}_t^e) \cdot \boldsymbol{W}^e) \tag{6}$$

$$\boldsymbol{h}_t^e = (1 - \boldsymbol{z}_t^e) \odot \boldsymbol{h}_{t-1}^e + \boldsymbol{z}_t^e \odot \tilde{\boldsymbol{h}}_t^e \tag{7}$$

where $\odot$ represents element-wise multiplication. $\boldsymbol{W}_z^e \in \mathbb{R}^{2d\times d}$, $\boldsymbol{W}_r^e \in \mathbb{R}^{2d\times d}$ and $\boldsymbol{W}^e \in \mathbb{R}^{d\times d}$ are trainable parameters. Att is a scaled key-value attention function (Vaswani et al., 2017), defined as:

$$\mathrm{Att}(\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v}) = \mathrm{Softmax}(\frac{(\boldsymbol{q}\boldsymbol{W}^q)(\boldsymbol{k}\boldsymbol{W}^k)^T}{\sqrt{d_a}})(\boldsymbol{v}\boldsymbol{W}^v) \tag{8}$$

where $\boldsymbol{W}^q \in \mathbb{R}^{d\times d_a}$, $\boldsymbol{W}^k \in \mathbb{R}^{d\times d_a}$ and $\boldsymbol{W}^v \in \mathbb{R}^{d\times d_a}$ are trainable parameters.

Then we update the location slots similarly:

$$\boldsymbol{z}_t^l = \sigma([\boldsymbol{h}_{t-1}^l; \boldsymbol{x}_t^l] \cdot \boldsymbol{W}_z^l) \tag{9}$$

$$\boldsymbol{r}_t^l = \sigma([\boldsymbol{h}_{t-1}^l; \boldsymbol{x}_t^l] \cdot \boldsymbol{W}_r^l) \tag{10}$$

$$\hat{\boldsymbol{h}}_t^l = [\boldsymbol{r}_t^l \odot \boldsymbol{h}_{t-1}^l; \boldsymbol{x}_t^l] \tag{11}$$

$$\tilde{\boldsymbol{h}}_t^l = \tanh(\mathrm{Att}(\hat{\boldsymbol{h}}_t^l, \boldsymbol{h}_t^e, \boldsymbol{h}_t^e) \cdot \boldsymbol{W}^l) \tag{12}$$

$$\boldsymbol{h}_t^l = (1 - \boldsymbol{z}_t^l) \odot \boldsymbol{h}_{t-1}^l + \boldsymbol{z}_t^l \odot \tilde{\boldsymbol{h}}_t^l \tag{13}$$

where $\boldsymbol{W}_z^l \in \mathbb{R}^{2d\times d}$, $\boldsymbol{W}_r^l \in \mathbb{R}^{2d\times d}$ and $\boldsymbol{W}^l \in \mathbb{R}^{d\times d}$ are trainable parameters. We initialize $\boldsymbol{h}_0^e$ and $\boldsymbol{h}_0^l$ using zero matrices.

The two key steps are Eq 6 and Eq 12. We use key-value attention to explicitly model the entity-entity interactions and the location-entity interactions. According to our intuition that an entity location is closely related by its most recent action, in Eq 12, we use $\boldsymbol{h}_t^e$, the newly-updated entity representation, as the input to the location-entity attention.

The newly updated entity and location representations $\boldsymbol{h}_t^e$ and $\boldsymbol{h}_t^l$ are then sent to different classifiers to predict the actions and locations, respectively, at this timestep.

Output 1: Entity Actions Following NCET (Gupta and Durrett, 2019), We use CRF to model the dependency of actions. We use $\boldsymbol{h}_t^{e_i}$, which is the ith row of $\boldsymbol{h}_t^e$, to generate emission potentials for each action tag y_t at each time step t with respect to entity e_i:

$$\phi(y_t, t, e_i) = \boldsymbol{h}_t^e \cdot \boldsymbol{H} \tag{14}$$

where $\boldsymbol{H} \in \mathbb{R}^{d\times 1}$ is a trainable matrix. Additionally, we train a transition matrix to get the transition potentials among the 4 action tags and two extra tags (”START” and ”END”) which we denote by $\psi(y_{t-1}, y_t)$. Finally for an action sequence $\boldsymbol{y}$, we get the probability as:

$$P(\boldsymbol{y}|e_i) \propto \exp(\sum_{t=0}^{T} \phi(y_t, t, e_i) + \psi(y_{t-1}, y_t)) \tag{15}$$

Output 2: Entity Locations To get the probability that entity e_i at location l_i at step t, we simply compute the probability matrix as follows:

$$\boldsymbol{M}_t = \mathrm{Softmax}((\boldsymbol{h}_t^e \boldsymbol{U}) \cdot (\boldsymbol{l}_t^e \boldsymbol{V})^T) \tag{16}$$

where $\boldsymbol{U} \in \mathbb{R}^{d\times d}$, $\boldsymbol{V} \in \mathbb{R}^{d\times d}$ are trainable parameter matrices. And the probability $P(e_i, l_j)$ is the element at the ith row and jth column of $\boldsymbol{M}_t$.

Training and Loss Function We jointly train the action prediction task and location prediction task, and the objective is to minimize the sum of their log likelihood losses. For each training step, we only use one paragraph. We use Adam (Kingma and Ba, 2014) with learning rate 0.0002 for training.

4.3 Postprocessing

Following previous conventions (Dalvi et al., 2018) to make the predictions consistent with commonsense (e.g., an entity cannot be destroyed if it is

Model	Document Level			Sentence Level				
	Precision	Recall	F1	Cat-1	Cat-2	Cat-3	Macro-Avg	Micro-avg
ProLocal	**81.7**	36.8	50.7	62.7	30.5	10.4	34.5	34.0
ProGlobal	48.8	**61.7**	51.9	63.0	36.4	35.9	45.1	45.4
ProStruct	74.3	43.0	54.5	-	-	-	-	-
LACE	75.3	45.4	56.6	-	-	-	-	-
KG-MRC	69.3	49.3	57.6	62.9	40.0	38.2	47.0	46.7
NCET*	71.0	51.1	59.4	70.4	44.3	40.6	51.7	51.2
NCET+ELMo*	70.9	53.7	61.1	71.1	46.6	**41.0**	52.9	52.4
IEN	69.5	55.1	61.4	71.2	45.6	40.0	52.6	52.1
IEN+ELMo	69.8	56.3	**62.3**	**71.8**	**47.6**	40.5	**53.3**	**53.0**

Table 1: Main results on ProPara. * represents our implementation, which we keep the preprocessing and postprocessing steps the same as IEN. For other methods, document level results are taken from the official leaderboard and sentence level results are taken from the original papers.

Model	Inputs			Outputs			Conversions			Moves			Overall		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
NCET*	78.6	55.7	65.1	78.7	76.8	77.7	**71.2**	36.6	48.2	55.3	35.4	43.0	**71.0**	51.1	59.4
NCET+ELMo*	**80.0**	61.0	69.0	**84.3**	76.3	80.0	66.3	41.2	50.6	53.1	36.3	43.0	70.9	53.7	61.1
IEN (no l2e)	76.8	60.0	67.3	78.6	**77.0**	77.8	64.5	45.1	53.0	55.4	34.5	42.4	68.9	54.2	60.6
IEN (parallel)	76.1	59.5	66.7	81.1	76.1	78.5	65.9	44.6	**53.1**	**60.0**	35.2	44.2	70.8	53.9	61.1
IEN	77.1	62.3	68.8	80.7	76.7	78.6	65.1	43.4	51.9	55.3	**37.7**	**44.8**	69.5	55.1	61.4
IEN+ELMo	79.2	**66.6**	**72.6**	84.0	75.3	**79.3**	63.1	**45.3**	52.7	52.7	37.4	43.6	69.8	**56.3**	**62.3**

Table 2: Detailed results on four kinds of questions asked by the document level evaluator.

not exist), we apply two kinds of rules to the predictions: 1) correct invalid actions according to the whole action sequence, and 2) locally correct the locations according to the corresponding actions.

5 Experiments

We evaluate our model on the two comprehension tasks of ProPara dataset using the official evaluator[2], and compare against competitive systems on the official leaderboard[3].

5.1 Evaluation Metrics

Document Level (Tandon et al., 2018) Given the predictions of a paragraph, the document level evaluation is to answer four templated questions, whose answers are deterministically computed from the predictions:
Q1: What are the inputs to the process?
Q2: What are the outputs of the process?
Q3: What conversions occur, when and where?
Q4: What movements occur, when and where?
Inputs are entities that exist at the start of the process, but not at the end. Outputs are entities that do not exist at the start, but do at the end. A conversion is when some entities are destroyed and others created. A movement is an event where an entity changes its location. For each paragraph, the evaluator compute a F1 score for each question, and the overall F1 score is the macro-average of the four questions.

Sentence Level (Dalvi et al., 2018) Given the predictions of an entity e in a paragraph, the sentence level task aims to answer 10 fine grained sentence level templated questions, which can be summarized into 3 categories:
Cat-1: Is e Created (Moved, Destroyed) in the process?
Cat-2: When was e Created (Moved, Destroyed)?
Cat-3: Where was e Created (Moved from/to, Destroyed)?
The evaluator automatically extracts the answer from the predictions and compute the accuracy for each question. Each category's accuracy is the macro-average of the containing questions, and the overall macro-average is the mean of the three categories' accuracy.

In this paper, we particularly concern about the ability of modeling entity interaction in event granularity (e.g., a conversion, which often contains more than one entities), and thus give more detailed results and analysis on document level evaluation.

[2]https://github.com/allenai/aristo-leaderboard/tree/master/propara
[3]https://leaderboard.allenai.org/propara/submissions/public

NCET:

water	oxygen	hydrogen	electrons
NONE plant → plant	CREATE - → ?	CREATE - → ?	CREATE - → ?

IEN:

water	oxygen	hydrogen	electrons
DESTROY plant → -	CREATE - → plant	CREATE - → plant	CREATE - → plant

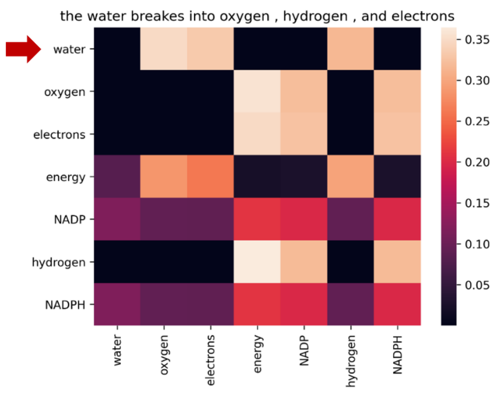


Figure 3: Case 1. The heatmap shows the entity-entity attention matrix for the example sentence.

5.2 Comparison Models

We compare our IEN with the following models:
ProLocal (Dalvi et al., 2018) predicts the state changes described in each individual sentence, and then uses commonsense rules of inertia to propagate state values forwards and backwards in time.
ProGlobal (Dalvi et al., 2018) predicts states of an entity across all timesteps by considering the entire paragraph.
ProStruct (Tandon et al., 2018) adopts a structured prediction method and incorporates a set of commonsense constraints for global consistency.
LACE (Du et al., 2019) leverages label consistency during training and allows consistency bias to be built into the model.
KG-MRC (Das et al., 2018) constructs dynamic bipartite graphs from the text, and updates the graphs at each step. It also extends a machine reading comprehension model for location extraction.
NCET (Gupta and Durrett, 2019) proposes a two-layer LSTM model to model state changes. Additionally, a neural CRF is used over the top to explicitly capture the constrains.

5.3 Results

As the size of ProPara is small and the variance cannot be ignored, we train NCET and IEN 10 times separately with different random seeds and take the average results.

Table 1 summarizes the main results. In the document level evaluation, ProLocal has the highest precision and ProGlobal gets the highest recall. Other models from the leaderboard use different methods to introduce constraints and get more balanced precision and recall scores, which lead to higher F1 scores. Among them, NCET achieves the previously best result with 59.4% in F1. Although NCET can well model the sequential context of a single entity, it ignores the context from another dimension, the interaction among different entities in the same event. We can see that IEN significantly outperforms all other models in the official leaderboard, achieving a new high in F1 (61.4%), 2% more than NCET, while getting the second place in recall (55.1%) and maintaining a relatively high precision (69.5%). This suggests that by exploiting the entity interactions, IEN is able to leverage a broader context for prediction, and thus correctly predicts more state changes compared to other methods. As (Gupta and Durrett, 2019) indicate that a pre-trained language model, e.g., ELMo (Peters et al., 2018) or BERT (Devlin et al., 2019), can help improve model performance when the training set is small, we apply ELMo to IEN, and obtain great improvement in recall, leading to 0.9% more in F1.

Table 2 shows the detailed results with respect to the four kinds of questions asked by the document level evaluator. Compared to NCET, although IEN has a lower precision, only 65.1%, to identify Conversions, it achieves a higher recall for Conversions, 6.8% more than NCET, resulting in a substantial increase in F1. Besides, we can see that IEN performs better than NCET on both Inputs and Outputs: IEN gets a higher recall for Inputs, with a 6.6% margin, and a higher precision to recognize Outputs. These two indicators suggest that more CREATE and DESTROY actions are correctly found in the process by IEN. We can see that IEN can better understand the role of each entity in a conversion by explicitly modeling the entity-entity interactions, thus can capture more accurate state changes that NCET may ignore.

As can be seen in columns Moves of Table 2, IEN outperforms NCET in recall by 2.3%. This is because by modeling location-entity interaction and the causal relationship of action and location, IEN is able to maintain the location information of each entity, and bring it to the next time step to prevent the absence of location information in

NCET:

blood	carbon dioxide	oxygen
MOVE heart → lung	NONE ? → ?	NONE ? → ?
NONE lung → lung	MOVE blood → blood	MOVE blood → blood
NONE lung → lung	MOVE blood → blood	MOVE blood → blood

IEN:

blood	carbon dioxide	oxygen
MOVE heart → lung	NONE ? → ?	NONE ? → ?
NONE lung → lung	MOVE lung → lung	NONE ? → ?
NONE lung → lung	NONE lung → lung	MOVE lung → blood

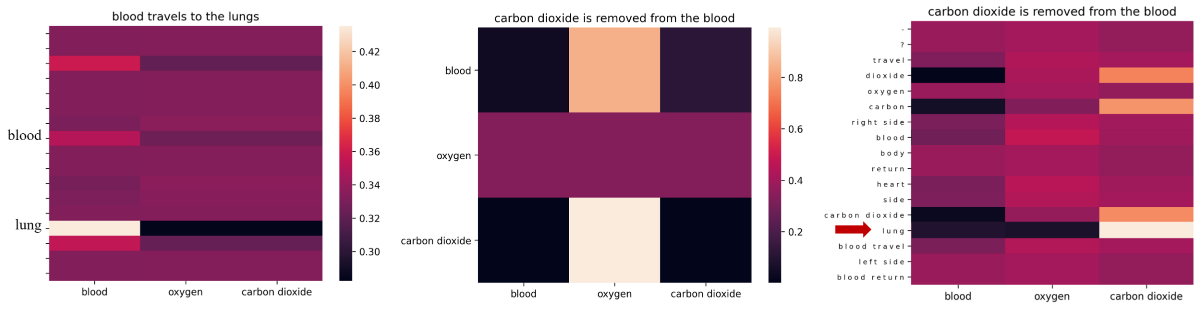


Figure 4: Case 2. Below are excerpts of three heatmaps, the left is the location-entity attention matrix of the first sentence, the middle and the right is the entity-entity attention matrix, and the location-entity attention matrix of the second sentence. There are some inconsistencies in the results, e.g., after the first step, oxygen locates at ”?”, but before the second step, oxygen locates at ”blood”. This is due to the rule-based postprocessing, which keeps local consistency, e.g., keeps the location before and after a step to be the same when this step's action is ”NONE”.

the sentence. We will use the ”carbon dioxide and lung” case in Sec. 5.5 for a detailed illustration.

As for sentence level evaluation, IEN also achieves higher overall macro- and micro- accuracy than the previous best method, NCET. More precisely, IEN gets higher Cat-1 and Cat-2 accuracy and competitive Cat-3 accuracy. The improvement of Cat-1 and Cat-2 accuracy can be attributed to the effectiveness of modeling entity interactions.

5.4 Ablation Analysis

We perform ablation studies to evaluate each component of IEN cells. We design two variants: 1) **IEN (no l2e)** tracks entity actions using simplified IEN cells that only contain entity slots and entity updating module, i.e., $\boldsymbol{h}_t^l$ and Eq 9 to Eq 13 are removed from the cells. For location prediction, we separately use BiLSTM like NCET. 2) **IEN (parallel)** contains IEN cells that run the entity updating module and location updating module in a parallel manner, i.e., in Eq 12, the input to location-entity attention module changes to $\hat{\boldsymbol{h}}_t^e$ instead of $\boldsymbol{h}_t^e$.

From table 2, compared to NCET, IEN (no l2e) performs well in the recall of Conversions, while the recall of Moves is similar to NCET and lower than IEN. This result confirms the effect of entity-entity interaction and location-entity interaction we discussed in the previous subsection.

As for IEN (parallel), even though this model explicitly models entity-entity interactions and location-entity interactions, the recall of Moves is lower than IEN by 2.5%, which indicates the importance of understanding the causal relationship between entity action and its location. As an entity's action is a decisive property, it is definitely helpful to have thorough and accurate information about the entity's action before figuring out its location.

5.5 Case Study

To better understand how the IEN works, here we conduct a case study using the examples in Sec. 1.

Figure 3 shows the results of NCET and IEN in the first case, in which 4 entities are involved in a chemical transformation. The key is to correctly identify the role of each entity in the conversion. Even though NCET correctly predicts the actions of oxygen, hydrogen, and electrons, it makes a wrong prediction on water, this is because NCET performs separate predictions for multiple entities in the same event. IEN models all entities and their interactions simultaneously, and perfectly predict all their actions. The heatmap in Figure 3 is the corresponding entity-entity attention matrix in the IEN cell. We find that the water row presents which entities have a higher probability of changing from water. Surprisingly, if we notice the oxygen (or hydrogen, electrons) row, the model even makes a *pre-filtering* about what the oxygen will become, which even appears before the actual description. We think this may be because that oxygen, hydrogen, and electrons are simultaneously come from water, thus are less likely to be the next product. Such pre-filtering mechanism may help future prediction by reducing the candidates. Besides, IEN correctly predicts all the locations which are not explicitly mentioned in the sentence. In fact, ”plant” comes from a previous sentence that describes ”the plant absorbs water from soil”. To pass the correct location from water to oxygen, the model needs first to identify which entity is the reactant in the chemical reaction, and then pass the reactant's location information to other entities. This shows that feeding up-to-date entity information for subsequent location prediction can help the model better understand what happens to the entity in the current action and know which states of the entity may change accordingly.

The second case is more complicated because it requires context reasoning across three sentences. As shown in Figure 4, similar to the case 1, NCET does not identify the role of all entities well, and confuses carbon dioxide with oxygen, resulting in redundant predictions for MOVEs, while IEN performs well on action prediction. As for predicting locations, IEN is able to correctly predict that, after removed from the blood, carbon dioxide locates at the lung, while the lung is actually mentioned in the previous sentence. IEN manages to capture the interaction between carbon dioxide and blood, and that between lung and blood in nearby sentences, making a reasoning chain from carbon dioxide to blood, and to lung. To see how IEN achieves this, we show the excerpts of 3 heatmaps in Figure 4. In the first step, IEN builds a connection between location lung and entity blood through location-entity interaction. At the second step, from the entity-entity attention matrix, we can see that IEN correctly finds out that carbon dioxide is moving apart from blood and thus gives a low attention weight. And from the location-entity attention matrix, we notice that carbon dioxide has correctly got the location information from the blood. Another interesting finding is that IEN can correctly predict that the oxygen is originally located at the lung even before reading the third sentence, where the oxygen first appears. This may be the combined effect of location-entity interactions and pre-filtering.

6 Conclusion

In this paper, we propose the Interactive Entity Network, IEN, for the multi-entity state tracking task, which learns to interpret complex processes by explicitly modeling the synergy among different entities involved in one event and leveraging the causal relationship between entity actions and their subsequent state changes. Experiments on ProPara show that IEN can better understand scientific procedural texts and outperforms state-of-the-art models.

Acknowledgements

This work is supported in part by the National Hi-Tech R&D Program of China (No. 2018YFC0831900), the NSFC Grants (No. 61672057, 61672058). For any correspondence, please contact Yansong Feng.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling (2014). *arXiv preprint arXiv:1412.3555*.

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 *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1595–1604, New Orleans, Louisiana. Association for Computational Linguistics.

Rajarshi Das, Tsendsuren Munkhdalai, Xingdi Yuan, Adam Trischler, and Andrew McCallum. 2018. Building dynamic knowledge graphs from text using machine reading comprehension. *arXiv preprint arXiv:1810.05682*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Xinya Du, Bhavana Dalvi, Niket Tandon, Antoine Bosselut, Wen-tau Yih, Peter Clark, and Claire

Cardie. 2019. Be consistent! improving procedural text comprehension using label consistency. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2347–2356, Minneapolis, Minnesota. Association for Computational Linguistics.

Aditya Gupta and Greg Durrett. 2019. Tracking discrete and continuous entity state for process understanding. *arXiv preprint arXiv:1904.03518*.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2016. Tracking the world state with recurrent entity networks. *arXiv preprint arXiv:1612.03969*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

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.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409, Austin, Texas. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Query-reduction networks for question answering. *arXiv preprint arXiv:1606.04582*.

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 *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 57–66, Brussels, Belgium. Association for Computational Linguistics.

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 *Advances in neural information processing systems*, pages 5998–6008.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.