Tracking State Changes in Procedural Text: A Challenge Dataset and Models for Process Paragraph Comprehension

Bhavana Dalvi Mishra[1*], **Lifu Huang**[2*], **Niket Tandon**[1], **Wen-tau Yih**[1], **Peter Clark**[1]
[1]Allen Institute for AI, Seattle, [2]Rensselaer Polytechnic Institute, Troy
{bhavanad,nikett,scottyih,peterc}@allenai.org, {warrior.fu}@gmail.com

Abstract

We present a new dataset and models for comprehending paragraphs about processes (e.g., photosynthesis), an important genre of text describing a dynamic world. The new dataset, ProPara, is the first to contain natural (rather than machine-generated) text about a changing world along with a full annotation of entity states (location and existence) during those changes (81k datapoints). The end-task, tracking the location and existence of entities through the text, is challenging because the causal effects of actions are often implicit and need to be inferred. We find that previous models that have worked well on synthetic data achieve only mediocre performance on ProPara, and introduce two new neural models that exploit alternative mechanisms for state prediction, in particular using LSTM input encoding and span prediction. The new models improve accuracy by up to 19%. The dataset and models are available to the community at `http://data.allenai.org/propara`.

1 Introduction

Building a reading comprehension (RC) system that is able to read a text document and to answer questions accordingly has been a long-standing goal in NLP and AI research. Impressive progress has been made in factoid-style reading comprehension, e.g., (Seo et al., 2017a; Clark and Gardner, 2017), enabled by well-designed datasets and modern neural network models. However, these models still struggle with questions that require *inference* (Jia and Liang, 2017).

Consider the paragraph in Figure 1 about photosynthesis. While top systems on SQuAD (Rajpurkar et al., 2016) can reliably answer lookup questions such as:
Q1: What do the roots absorb? (A: water, minerals)
they struggle when answers are not explicit, e.g.,
Q2: Where is sugar produced? (A: in the leaf)[1]

> Chloroplasts in the leaf of the plant trap light from the sun. The roots absorb water and minerals from the soil. This combination of water and minerals flows from the stem into the leaf. Carbon dioxide enters the **leaf**. Light, water and minerals, and the carbon dioxide all combine into a mixture. This mixture forms **sugar** (glucose) which is what the plant eats.
>
> **Q:** Where is sugar produced?
> **A:** in the leaf

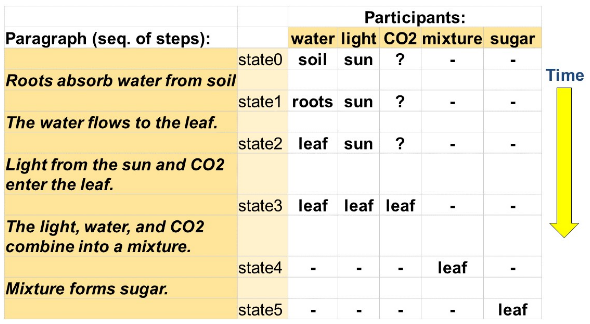
Figure 1: A paragraph from *ProPara* about photosynthesis (bold added, to highlight question and answer elements). Processes are challenging because questions (e.g., the one shown here) often require inference about the process states.

To answer Q2, it appears that a system needs knowledge of the world and the ability to reason with state transitions in multiple sentences: If carbon dioxide *enters* the leaf (stated), then it will be *at* the leaf (unstated), and as it is then used to produce sugar, the sugar production will be at the leaf too.

This challenge of modeling and reasoning with a changing world is particularly pertinent in text about *processes*, demonstrated by the paragraph in Figure 1. Understanding what is happening in such texts is important for many tasks, e.g., procedure execution and validation, effect prediction. However, it is also difficult because the world state is changing, and the causal effects of actions on that state are often implicit.

To address this challenging style of reading comprehension problem, researchers have created several datasets. The bAbI dataset (Weston et al., 2015) includes questions about objects moved throughout a paragraph, using machine-generated language over a deterministic domain with a small lexicon. The SCoNE dataset (Long et al., 2016) contains paragraphs describing a changing world state in three synthetic, deterministic domains, and

[*]Bhavana Dalvi Mishra and Lifu Huang contributed equally to this work.
[1]For example, the RC system BiDAF (Seo et al., 2017a) answers "glucose" to this question.



		Participants:				
Paragraph (seq. of steps):		water	light	CO2	mixture	sugar
	state0	soil	sun	?	-	-
Roots absorb water from soil						
	state1	roots	sun	?	-	-
The water flows to the leaf.						
	state2	leaf	sun	?	-	-
Light from the sun and CO2 enter the leaf.						
	state3	leaf	leaf	leaf	-	-
The light, water, and CO2 combine into a mixture.						
	state4	-	-	-	leaf	-
Mixture forms sugar.						
	state5	-	-	-	-	leaf

Figure 2: A (simplified) annotated paragraph from ProPara. Each filled row shows the existence and location of participants between each step ("?" denotes "unknown", "-" denotes "does not exist"). For example in state0, water is located at the soil.

assumes that a complete and correct model of the initial state is given for each task. However, approaches developed using synthetic data often fail to handle the inherent complexity in language when applied to organic, real-world data (Hermann et al., 2015; Winograd, 1972).

In this work, we create a new dataset, *ProPara* (Process Paragraphs), containing 488 human-authored paragraphs of procedural text, along with 81k annotations about the changing states (existence and location) of entities in those paragraphs, with an end-task of predicting location and existence changes that occur. This is the first dataset containing annotated, natural text for real-world processes, along with a simple representation of entity states during those processes. A simplified example is shown in Figure 2.

When applying existing state-of-the-art systems, such as Recurrent Entity Networks (Henaff et al., 2016) and Query-reduction Networks (Seo et al., 2017b), we find that they do not perform well on *ProPara* and the results are only slightly better than the majority baselines. As a step forward, we propose two new neural models that use alternative mechanisms for state prediction and propagation, in particular using LSTM input encoding and span prediction. The new models improve accuracy by up to 19%.

Our contributions in this work are twofold: (1) we create *ProPara*, a new dataset for process paragraph comprehension, containing annotated, natural language paragraphs about real-world processes, and (2) we propose two new models that learn to infer and propagate entity states in novel ways, and outperform existing methods on this dataset.

2 Related Work

Datasets: Large-scale reading comprehension datasets, e.g., SQuAD (Rajpurkar et al., 2016), TriviaQA (Joshi et al., 2017), have successfully driven progress in question answering, but largely targeting explicitly stated facts. Often, the resulting systems can be fooled (Jia and Liang, 2017), prompting efforts to create harder datasets where a deeper understanding of the text appears necessary (Welbl et al., 2017; Araki et al., 2016).

Procedural text is a genre that is particularly challenging, because the worlds they describe are largely implicit and changing. While there are few large datasets in this genre, two exceptions are bAbI (Weston et al., 2015) and SCoNE (Long et al., 2016), described earlier[2]. bAbI has helped advance methods for reasoning over text, such as memory network architectures (Weston et al., 2014), but has also been criticized for using machine-generated text over a simulated domain. SCoNE is closer to our goal, but has a different task (*given* a perfect world model of the initial state, predict the end state) and different motivation (handling ellipsis and coreference in context). It also used a deterministic, simulated world to generate data.

Models: For answering questions about procedural text, early systems attempted to extract a process structure (events, arguments, relations) from the paragraph, e.g., ProRead (Berant et al., 2014) and for newswire (Caselli et al., 2017). This allowed questions about event ordering to be answered, but not about state changes, unmodelled by these approaches.

More recently, several neural systems have been developed to answer questions about the world state after a process, inspired in part by the bAbI dataset. Building on the general Memory Network architecture (Weston et al., 2014) and gated recurrent models such as GRU (Cho et al., 2014), Recurrent Entity Networks (EntNet) (Henaff et al., 2016) is a state-of-the-art method for bAbI. EntNet uses a dynamic memory of hidden states (memory blocks) to maintain a representation of the world state, with a gated update at each step. Memory keys can be preset ("tied") to particular entities in the text, to encourage the memories to record information about those entities. Similarly, Query Reduction Networks (QRN) (Seo et al., 2017b) tracks state in

[2]The ProcessBank (Berant et al., 2014) dataset is smaller and does not address state change, instead containing 585 questions about event ordering and event arguments.

a paragraph, represented as a hidden vector h. QRN performs gated propagation of h across each timestep (corresponding to a state update), and uses h to modify ("reduce") the query to keep pointing to the answer at each step (e.g., "Where is the apple?" at step_1 might be modified to "Where is Joe?" at step_2 if Joe picks up the apple). A recent proposal, Neural Process Networks (NPN) (Bosselut et al., 2018), also models each entity's state as a vector (analogous to EntNet's tied memories). NPN computes the state change at each step from the step's predicted action and affected entity(s), then updates the entity(s) vectors accordingly, but does not model different effects on different entities by the same action.

Both EntNet and QRN find a final answer by decoding the final vector(s) into a vocabulary entry via softmax classification. In contrast, many of the best performing factoid QA systems, e.g., (Seo et al., 2017a; Clark and Gardner, 2017), return an answer by finding a *span* of the original paragraph using attention-based span prediction, a method suitable when there is a large vocabulary. We combine this span prediction approach with state propagation in our new models.

3 The ProPara Dataset

Task: Our dataset, *ProPara*, focuses on a particular genre of procedural text, namely simple scientific processes (e.g., photosynthesis, erosion). A system that understands a process paragraph should be able to answer questions such as: "*What are the inputs to the process?*", "*What is converted into what?*", and "*Where does the conversion take place?*"[3] Many of these questions reduce to understanding the basic dynamics of entities in the process, and we use this as our task: Given a process paragraph and an entity e mentioned in it, identify:
(1) **Is** e created (destroyed, moved) in the process?
(2) **When** (step #) is e created (destroyed, moved)?
(3) **Where** is e created (destroyed, moved from/to)?
If we can track the entities' *states* through the process and answer such questions, many of the higher-level questions can be answered too. To do this, we now describe how these states are representated in ProPara, and how the dataset was built.

Process State Representation: The states of the world throughout the whole process are represented as a grid. Each column denotes a *participant* entity (a span in the paragraph, typically a noun phrase) that undergoes some creation, destruction, or movement in the process. Each row denotes the *states* of all the participants after a *step*. Each sentence is a step that may change the state of one or more participants. Therefore, a process paragraph with m sentences and n participants will result in an $(m + 1) \times n$ grid representation. Each cell l_{ij} in this grid records the *location* of the j-th participant after the i-th step, and l_{0j} stores the location of j-th participant before the process.[4] Figure 2 shows one example of this representation.

Paragraph Authoring: To collect paragraphs, we first generated a list of 200 process-evoking prompts, such as "*What happens during photosynthesis?*", by instantiating five patterns[5], with nouns of the corresponding type from a science vocabulary, followed by manual rewording. Then, crowdsourcing (MTurk) workers were shown one of the prompts and asked to write a sequence of event sentences describing the process. Each prompt was given to five annotators to produce five (independent) paragraphs. Short paragraphs (4 or less sentences) were then removed for a final total of 488 paragraphs describing 183 processes. An example paragraph is the one shown earlier in Figure 1.

Grid and Existence: Once the process paragraphs were authored, we asked expert annotators[6] to create the initial grids. First, for each paragraph, they listed the participant entities that underwent a state change during the process, thus creating the column headers. They then marked the steps where a participant was created or destroyed. All state cells before a Create or after a Destroy marker were labeled as "not exists". Each initial grid annotation was checked by a second expert annotator.

Locations: Finally, MTurk workers were asked to fill in all the location cells. A location can be "unknown" if it is not specified in the text, or a span of the original paragraph. Five grids for the same paragraph were completed by five different Turkers, with average pairwise inter-annotator agreement of 0.67. The end result was 81,345 annotations over 488 paragraphs about 183 processes. The dataset

[3]For example, science exams pose such questions to test student's understanding of the text in various ways.

[4]We only trace locations in this work, but the representation can be easily extended to store other properties (e.g., temperature) of the participants.

[5]The five patterns are: How are ***structure*** formed? How does ***system*** work? How does ***phenomenon*** occur? How do you use ***device***? What happens during ***process***?

[6]Expert annotators were from our organization, with a college or higher degree.

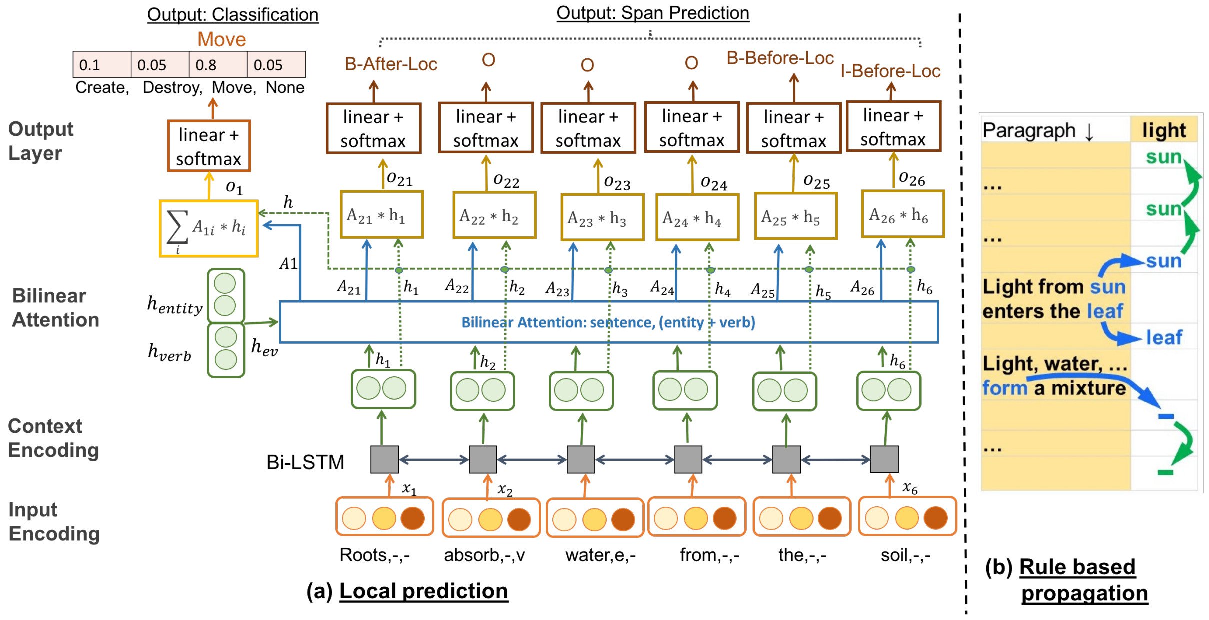


Figure 3: (a) ProLocal uses bidirectional attention to make local predictions about state change type and location (left), and then (b) propagates those changes globally using a persistence rule (right, shown for a single participant (the Light), local predictions shown in blue, propagations via persistence in green).

	bAbI	SCoNE	ProPara
Sentences	Synthetic	Natural	Natural
Questions	templated	templated	templated
# domains	20	3	183
Vocab #words	119	1314	2501
# sentences	131.1k	72.9k	3.3k
# unique sents	3.2k	37.4k	3.2k
Avg words/sent	6.5	10.2	9.0

Table 1: ProPara vs. other procedural datasets.

was then split 80/10/10 into train/dev/test by *process prompt*, ensuring that the test paragraphs were all about processes unseen in train and dev. Table 1 compares our dataset with bAbI and SCoNE.

4 Models

We present two new models for this task. The first, ProLocal, makes local state predictions and then algorithmically propagates them through the process. The second, ProGlobal, is an end-to-end neural model that makes all state predictions using global information.

4.1 ProLocal: A Local Prediction Model

The design of ProLocal consists of two main components: *local prediction* and *commonsense persistence*. The former infers all direct effects of individual sentences and the latter algorithmically propagates known values forwards and backwards to fill in any remaining unknown states.

4.1.1 Local Prediction

The intuition for local prediction is to treat it as a surface-level QA task. BiLSTMs with span prediction have been effective at answering surface-level questions, e.g., Given "Roots absorb water." and "Where is the water?", they can be reliably trained to answer "Roots" (Seo et al., 2017a). We incorporate a similar mechanism here.

Given a sentence (step) and a participant e in it, the local prediction model makes two types of predictions: the change type of e (one of: *no change*, *created*, *destroyed*, *moved*) and the locations of e before and after this step. The change type prediction is a multi-class classification problem, while the location prediction is viewed as a SQuAD-style surface-level QA task with the goal to find a location span in the input sentence. The design of this model is depicted in Figure 3(a), which adapts a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) recurrent neural network architecture (biLSTM) with attention for input encoding. The prediction tasks are handled by two different output layers. We give the detail of these layers below.

Input Encoding: Each word w_i in the input sentence is encoded with a vector $x_i = [v_w : v_e : v_v]$, the concatenation of a pre-trained GloVe (Pennington et al., 2014) word embedding v_w, indicator variables v_e on whether w_i is the specified participant and v_v on whether w_i is a verb (via a POS tagger).

Context Encoding: A biLSTM is used to contextualize the word representations in a given sentence. h_i denotes the concatenated output of the bidirectional LSTM for the embedded word x_i, and encodes the word's meaning in context.

Bilinear Attention: Given the participant and verb, the role of this layer is to identify which contextual word embeddings to attend to for generating the output. We first create h_{ev} by concatenating the contextual embedding of the participant and verb.[7] We then use a bilinear similarity function $sim(h_i, h_{ev}) = (h_i^T * B * h_{ev}) + b$, similar to (Chen et al., 2016), to compute attention weights A_i over each word w_i in the sentence.

For state change type prediction, the words between the verb and participant may be important, while for the location tagging, contextual cues such as "from" and "to" could be more predictive. Hence, we train two sets of attention parameters resulting in weights A_1 and A_2 which are combined with the contextual vectors h_i as described below to produce hidden states o_1 and o_2 that are fed to the output layers. Here, $|step|$ refers to number of words in the given step or sentence.

$$o_1 = \sum_i A_{1i} * h_i$$

$$o_2 = [(A_{21} * h_1) : (A_{22} * h_2) : \ldots : (A_{2|step|} * h_{|step|})]$$

Output 1: State Change Type: We apply a feed-forward network on hidden state o_1 to derive the probabilities of the four state change type categories: Create, Destroy, Move and None.

Output 2: Location Spans: The second output is computed by predicting BIO tags (one of five tags: B-Before-LOC, I-Before-LOC, B-After-LOC, I-After-LOC, O) for each word in the sentence. We apply a feed-forward network on hidden state o_{2i} for $word_i$ to derive the probabilities of these location tags. Notice that if the change type is predicted as "Create" (or "Destroy") then only the "after" (or "before") location prediction is used.

Training: We train the state change type prediction and location tag prediction models jointly, where the loss is the sum of their negative log likelihood losses. We use Adadelta (Zeiler, 2012) with learning rate 0.2 to minimize the total loss.

4.1.2 Commonsense Persistence

The local prediction model will partially fill in the state change grid, showing the direct locational effects of actions (including "not exists" and "unknown location"). To complete the grid, we then algorithmically apply a commonsense rule of persistence that propagates locations forwards and backwards in time where locations are otherwise missing. Figure 3(b) shows an example when applying this rule, where '?' indicates "unknown location". This corresponds to a rule of inertia: things are by default unchanged unless told otherwise. If there is a clash, then the location is predicted as unknown.

4.2 ProGlobal: A Global Prediction Model

Unlike ProLocal, the design principle behind ProGlobal is to model the persistence of state information *within* the neural model itself, rather than as a post-processing step. ProGlobal infers the states of *all* participants at each step, even if they are not mentioned in the current sentence, using: (1) the global context (i.e., previous sentences), and (2) the participant's state from the previous step.

Given a sentence (step) with its context (paragraph) and a participant e, ProGlobal predicts the existence and location of e after this step in two stages. It first determines the state of e as one of the classes ("not exist", "unknown location", "known location"). A follow-up location span prediction is made if the state is classified as "known location".

Figure 4 shows ProGlobal's neural architecture, where the left side is the part for state prediction at each step, and the right side depicts the propagation of hidden states from one step to the next. We discuss the detail of this model below.

Input Encoding: Given a participant e, for each step$_i$, we take the entire paragraph as input. Each word w in the paragraph is represented with three types of embeddings: the general word embedding v_w, a position embedding v_d which indicates the relative distance to the participant in the paragraph, and a sentence indicator embedding v_s which shows the relative position (*previous, current, following*) of each sentence in terms of the current step i. Both the position embedding and the sentence indicator embedding are of size 50 and are randomly initialized and automatically trained by the model. We concatenate these three types of embeddings to represent each word $x = [v_w : v_d : v_s]$.

Context Encoding: Similar to ProLocal, we use a biLSTM to encode the whole paragraph and use $\tilde{h}$ to denote the biLSTM output for each word.

State Prediction: As discussed earlier, we first predict the location state of a participant e. Let

[7]Multi-word entities/verbs or multiple verbs are represented by the average word vectors.

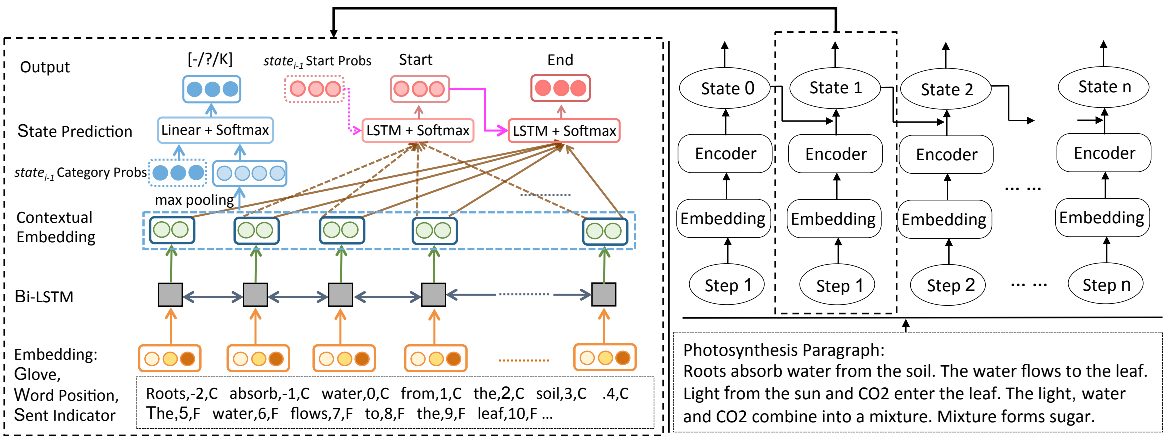

Output
[-/?/K]
state_{i-1} Start Probs
Start
End
State Prediction
Linear + Softmax
LSTM + Softmax
LSTM + Softmax
state_{i-1} Category Probs
max pooling
Contextual Embedding
Bi-LSTM
Embedding: Glove, Word Position, Sent Indicator
Roots,-2,C absorb,-1,C water,0,C from,1,C the,2,C soil,3,C .4,C
The,5,F water,6,F flows,7,F to,8,F the,9,F leaf,10,F ...
State 0
State 1
State 2
State n
Encoder
Embedding
Step 1
Step 2
Step n
Photosynthesis Paragraph:
Roots absorb water from the soil. The water flows to the leaf. Light from the sun and CO2 enter the leaf. The light, water and CO2 combine into a mixture. Mixture forms sugar.

Figure 4: ProGlobal predicts a participant's state (type and location) after a given step using bilinear attention over the entire paragraph, combined with its predictions from the previous step.

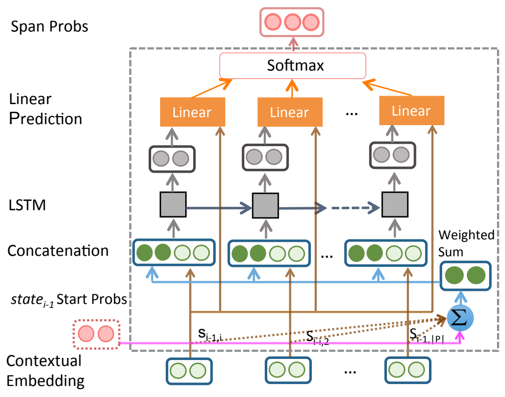


Figure 5: Details of the LSTM+Softmax unit, used for predicting the start/end words of a location.

$\tilde{H}_i^P = [\tilde{h}_i^1, \tilde{h}_i^2, ..., \tilde{h}_i^{|P|}]$ denote the hidden vectors (contextual embeddings) for words in step$_i$ with respect to participant e, where h_i^t denotes the t-th word representation output by the biLSTM layer and P is the whole paragraph. We then apply max pooling to derive a paragraph representation: $\mu_i^P = \max(\tilde{H}_i^P)$. To incorporate the category prediction of the previous step, step$_{i-1}$, we concatenate its probability vector $c_{i-1}^P \in \mathbb{R}^3$ with μ_i^P, and apply a feed-forward network to derive the probabilities of the three categories:

$$c_i^P = \text{softmax}(W_c \cdot [\mu_i^P : c_{i-1}^P] + b_c)$$

Location Span Prediction: (Figure 5). To predict the location span, we predict the start word of the span (by generating a probability distribution over words) and the end word. To predict the location start, we take two types of information as input: the start probability distribution $s_{i-1}^P \in \mathbb{R}^{|P|}$ predicted from step$_{i-1}$, and the contextual embeddings $\tilde{H}_i^P$ of words in the current step$_i$:

$$\tilde{H}_i^* = \sum_{t=1}^{|P|} s_{i-1}^t \cdot \tilde{H}_i^t$$

$$\varphi_i^t = \text{LSTM}([\tilde{H}_i^t : \tilde{H}_i^*])$$

where $\tilde{H}_i^*$ is a sum of word vectors in the paragraph, weighted by the start probabilities from the previous step step$_{i-1}$. φ_i^t is the encoded vector representation for the t-th word in the paragraph. We then concatenate $\tilde{H}_i^P$ and φ_i^P, and apply a feed-forward network to obtain the start probability distribution for step$_i$: $s_i^P = \text{softmax}(W_s \cdot [\tilde{H}_i^P : \varphi_i^P] + b_s)$. Similarly, to predict the end word of the span, we use the start probability distribution s_i^P of *step*$_i$ and $\tilde{H}_i^P$, and apply another LSTM and feed-forward networks to obtain the probabilities. For state$_0$ (the initial location before any steps), we directly feed the sequence of the vectors from the encoding layer to a linear transformation to predict the location start, and apply the same architecture to predict the location end.

Training: For each participant e of paragraph P, the objective is to optimize the sum of the negative log likelihood of the category classification and location span prediction[8]. We use Adadelta to optimize the models with learning rate 0.5.

5 Experiments and Analysis

5.1 Tasks & Evaluation Metrics

As described in Section 3, the quality of a model is evaluated based on its ability to answer three categories of questions, with respect to a given participant e:

[8]We compute the loss for location span prediction only when the category is annotated as "known location".

	Sentence Encoding	Intermediate State Representn.	Propagation through time	Answer Decoding
EntNet	positional encoding	Dynamic memory blocks	Gated propagation	Softmax classification
QRN	positional encoding	Single latent vector h	Gated propagation of h	Softmax classification
ProLocal	LSTM	Explicit symbolic	Algorithmic	Span prediction
ProGlobal	LSTM	Distribution over spans	LSTM	Span prediction

Table 2: Design decisions in the four neural models.

(Cat-1) **Is** *e* created (destroyed, moved) in the process?
(Cat-2) **When** (step#) is *e* created (destroyed, moved)?
(Cat-3) **Where** is *e* created (destroyed, moved from/to)?
These questions are answered by simple scans over the state predictions for the whole process. (Cat-1) is asked over all participants, while (Cat-2) and (Cat-3) are asked over just those participants that were created (destroyed, moved). The accuracy of the answers is used as the evaluation metric, except for questions that may have multiple answers (e.g., "*When is e moved?*"). In this case, we compare the predicted and gold answers and use the F_1 score as the "accuracy" of the answer set prediction.[9]

For questions in category (3), an answer is considered correct if the predicted location is identical to, or a sub-phrase of, the labeled location (typically just one or two words), after stop-word removal and lemmatizing.

5.2 Baseline Methods

We compare our models with two top methods inspired by the bAbI dataset, Recurrent Entity Networks (EntNet) and Query Reduction Networks (QRN), described earlier in Section 2. Both models make different use of gated hidden states to propagate state information through time, and generate answers using softmax. The detailed comparisons in their design are shown in Table 2.

We use the released implementations[10] (with default hyper-parameter values), and retrained them on our dataset, adapted to the standard bAbI QA format. Specifically, we create three separate variations of data by adding three bAbI-style questions after each step in a paragraph, respectively:

Q1. "Does *e* exist?" (yes/no)
Q2. "Is the location of *e* known?" (yes/no)
Q3. "Where is *e*?" (span)

The template Q1 is applied to all participants. Q2 will only be present in the training data if Q1 is "yes", and similarly Q3 is only present if Q2 is "yes". These three variations of data are used to train three different models from the same method.

At test time, we cascade the questions (e.g., ask Q2 only if the answer to the Q1 model is "yes"), and combine the model outputs accordingly to answer the questions in our target tasks (Section 5.1).

We also compare against a rule-based baseline and a feature-based baseline. The rule-based method, called ProComp, uses a set of rules that map (a SRL analysis of) each sentence to its effects on the world state, e.g., IF X moves to Y THEN after: at(X,Y). The rules were extracted from VerbNet (Schuler, 2005) and expanded. A full description of ProComp is available at (Clark et al., 2018). The feature-based method uses a Logistic Regression (LR) classifier to predict the state change type (Move, Create, etc.) for each participant + sentence pair, then a NER-style CRF model to predict the from/to locations as spans of the sentence. The LR model uses bag-of-word features from the sentence, along with a discrete feature indicating whether the participant occurs before or after the verb in the given sentence. The CRF model uses standard NER features including capitalization, a verb indicator, the previous 3 words, and the POS-tag of the current and previous word. Similar to our ProLocal model, we apply commonsense persistence rules (Section 4.1.2) to complete the partial state-change grids predicted by both these baselines.

5.3 Results

Parameter settings: Both our models use GloVe embeddings of size 100 pretrained on Wikipedia 2014 and Gigaword 5 corpora[11]. The number of hidden dimensions for the biLSTM are set to 50(ProLocal) and 100(ProGlobal). Dropout rates (Srivastava et al., 2014) for the contextual encoding layer are 0.3(ProLocal) and 0.2(ProGlobal). ProGlobal uses word position and sentence indicator embeddings each of size 50, and span prediction LSTMs with a hidden dimension of 10. The learning rates for Adadelta optimizer were

[9]This approach has been adopted previously for questions with multiple answers (e.g., (Berant et al., 2013)). For questions with only one answer, F_1 is equivalent to accuracy.

[10]`https://github.com/siddk/entity-network` and `https://github.com/uwnlp/qrn`

[11]`https://nlp.stanford.edu/projects/glove`

Question type	Baseline Models					Our Models		Human
(# questions)	Majority	QRN	EntNet	Rule-based	Feature-based	ProLocal	ProGlobal	Upper Bound
Cat-1 (750)	51.01	52.37	51.62	57.14	58.64	62.65	62.95	91.67
Cat-2 (601)	-	15.51	18.83	20.33	20.82	30.50	36.39	87.66
Cat-3 (823)	-	10.92	7.77	2.4	9.66	10.35	35.9	62.96
macro-avg	-	26.26	26.07	26.62	29.7	34.50	45.08	80.76
micro-avg		26.49	25.96	26.24	29.64	33.96	45.37	79.69

Table 3: Model accuracy on the end task (test partition of *ProPara*). Questions are (Section 5.1): (Cat-1) **Is** $\boldsymbol{e_i}$ created (destroyed, moved)? (Cat-2) **When** is $\boldsymbol{e_i}$ created (...)? (Cat-3) **Where** is $\boldsymbol{e_i}$ created (...)?

0.2(ProLocal) and 0.5(ProGlobal). Our models are trained on the train partition and the parameters tuned on the dev partition.

Table 3 compares the performance of various models on the *ProPara* test partition. For the first category of questions, we also include a simple majority baseline. We aggregate results over the questions in each category, and report both macro and micro averaged accuracy scores.

From Table 3, we can see that EntNet and QRN perform comparably when applied to *ProPara*. However, despite being the top-performing systems for the bAbI task, when predicting whether a participant entity is created, destroyed or moved, their predictions are only slightly better than the majority baseline. Compared to our local model ProLocal, EntNet and QRN are worse in predicting the exact step where a participant is created, destroyed or moved, but better in predicting the location. The weak performance of EntNet and QRN on *ProPara* is understandable: both systems were designed with a different environment in mind, namely a large number of examples from a few conceptual domains (e.g., moving objects around a house), covering a limited vocabulary. As a result, they might not scale well when applied to real procedural text, which justifies the importance of having a real challenge dataset like *ProPara*.

Although the rule-based baseline (Clark et al., 2018) uses rules mapping SRL patterns to state changes, its performance appears limited by the incompleteness and approximations in the rulebase, and by errors by the SRL parser. The feature-based baseline performs slightly better, but its performance is still poor compared to our neural models. This suggests that it has not generalized as well to unseen vocabulary (25% of the test vocabulary is not present in the train/dev partitions of *ProPara*).

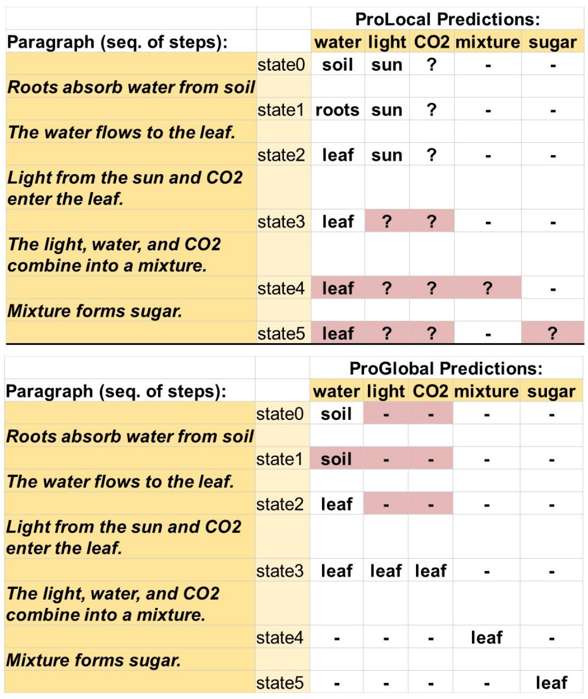
When comparing our two models, it is interesting that ProGlobal performs substantially better than ProLocal. One possible cause of this is cascading errors in ProLocal: if a local state prediction is wrong, it may still be propagated to later time steps without any potential for correction, thus amplifying the error. In contrast, ProGlobal makes a state decision for every participant entity at every time-step, taking the global context into account, and thus appears more robust to cascading errors. Furthermore, ProGlobal's gains are mainly in Cat-2 and Cat-3 predictions, which rely more heavily on out-of-sentence cues. For example, 30% of the time the end-location is not explicitly stated in the state-change sentence, meaning ProLocal cannot predict the end-location in these cases (as no sentence span contains the end location). ProGlobal, however, uses the entire paragraph and may identify a likely end-location from earlier sentences.

Finally, we computed a human upper bound for this task (last column of Table 3). During dataset creation, each grid was fully annotated by 5 different Turkers (Section 3). Here, for each grid, we identify the Turker whose annotations result in the best score for the end task with respect to the other Turkers' annotations. The observed upper bound of ~80% suggests that the task is both feasible and well-defined, and that there is still substantial room for creating better models.

5.4 Analysis

To further understand the strengths and weaknesses of our systems, we ran the simplified paragraph in Figure 2 verbatim through the models learned by ProLocal and ProGlobal. The results are shown in Figure 6, with errors highlighted in red.

ProLocal correctly interprets "*Light from the sun and CO2 enters the leaf.*" to imply that the light was at the sun before the event. In addition, as there were no earlier mentions of light, it propagates this location backwards in time, (correctly) concluding the light was initially at the sun. However, it fails to predict that "*combine*" (after state 3) destroys the inputs, resulting in continued prediction of the existence and locations for those inputs. One contributing factor is that ProLocal's predic-



Paragraph (seq. of steps):		ProLocal Predictions: water	light	CO2	mixture	sugar
	state0	soil	sun	?	-	-
Roots absorb water from soil						
	state1	roots	sun	?	-	-
The water flows to the leaf.						
	state2	leaf	sun	?	-	-
Light from the sun and CO2 enter the leaf.						
	state3	leaf	?	?	-	-
The light, water, and CO2 combine into a mixture.						
	state4	leaf	?	?	?	-
Mixture forms sugar.						
	state5	leaf	?	?	-	?

Paragraph (seq. of steps):		ProGlobal Predictions: water	light	CO2	mixture	sugar
	state0	soil	-	-	-	-
Roots absorb water from soil						
	state1	soil	-	-	-	-
The water flows to the leaf.						
	state2	leaf	-	-	-	-
Light from the sun and CO2 enter the leaf.						
	state3	leaf	leaf	leaf	-	-
The light, water, and CO2 combine into a mixture.						
	state4	-	-	-	leaf	-
Mixture forms sugar.						
	state5	-	-	-	-	leaf

Figure 6: ProLocal (top) and ProGlobal (bottom) predictions on a simple paragraph (errors in red).

tions ignore surrounding sentences (context), potentially making it harder to distinguish destructive vs. non-destructive uses of "*combine*".

ProGlobal also makes some errors on this text, most notably not realizing the light and CO2 exist from the start (rather, they magically appear at the leaf). Adding global consistency constraints may help avoid such errors. It is able to predict the sugar is formed at the leaf, illustating its ability to persist and transfer location information from earlier sentences to draw correct conclusions.

We additionally randomly selected 100 prediction errors from the dev set for ProGlobal, and identified four phenomena contributing to errors:

(1) Implicit Creation/Destruction: In 37% of the errors, the information about the creation or destruction of a participant is implicit or missing, which resulted in existence classification errors. For example, in the sentences "*A fuel goes into the generator. The generator converts mechanical energy into electrical energy.*", "*fuel*" is implicitly consumed as the generator converts mechanical energy into electrical energy.

(2) Location Errors: In 27% of the examples, the location spans were not perfectly identified as follows: absolute wrong location span prediction (17%), longer span prediction (6%), and location prediction from different granularity (4%).

(3) Complex Syntax: In 13% of the examples, a moving participant and its target location are separated with a wide context within a sentence, making it harder for the model to locate the location span.

(4) Propagation: ProGlobal tends to propagate the previous location state to next step, which may override locally detected location changes or propagate the error from previous step to next steps. 9% of the errors are caused by poor propagation.

5.5 Future Directions

This analysis suggests several future directions:
Enforcing global consistency constraints: e.g., it does not make sense to create an already-existing entity, or destroy a non-existent entity. Global constraints were found useful in the earlier ProRead system (Berant et al., 2014).
Data augmentation through weak supervision: additional training data can be generated by applying existing models of state change, e.g., from VerbNet (Kipper et al., 2008), to new sentences to create additional sentence+state pairs.
Propagating state information backwards in time: if e_j is at l_{ij} after $step_i$, it is likely to also be there at $step_{i-1}$ given no information to the contrary. ProGlobal, EntNet, and QRNs are inherently unable to learn such a bias, given their forward-propagating architectures.

6 Conclusion

New datasets and models are required to take reading comprehension to a deeper level of machine understanding. As a step in this direction, we have created the ProPara dataset, the first to contain natural text about a changing world along with an annotation of entity states during those changes. We have also shown that this dataset presents new challenges for previous models, and presented new models that exploit ideas from surface-level QA, in particular LSTM input encoding and span prediction, producing performance gains. The dataset and models are available at `http://data.allenai.org/propara`.

Acknowledgements

We are grateful to Paul Allen whose long-term vision continues to inspire our scientific endeavors. We also thank Oren Etzioni, Carissa Schoenick, Mark Neumann, and Isaac Cowhey for their critical contributions to this project.

References

Jun Araki, Dheeraj Rajagopal, Sreecharan Sankaranarayanan, Susan Holm, Yukari Yamakawa, and Teruko Mitamura. 2016. Generating questions and multiple-choice answers using semantic analysis of texts. In *COLING*. pages 1125–1136.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proc. EMNLP'13*.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D Manning. 2014. Modeling biological processes for reading comprehension. In *Proc. EMNLP'14*.

Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. 2018. Simulating action dynamics with neural process networks. *6th International Conference on Learning Representations (ICLR)* .

Tommaso Caselli, Ben Miller, Marieke van Erp, Piek Vossen, Martha Palmer, Eduard Hovy, Teruko Mitamura, and David Caswell. 2017. Proceedings of the events and stories in the news workshop. In *Proceedings of the Events and Stories in the News Workshop*.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. *CoRR* abs/1606.02858.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proc. Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST'14)*. pages 103–111.

Christopher Clark and Matt Gardner. 2017. Simple and effective multi-paragraph reading comprehension. *arXiv preprint arXiv:1710.10723* .

Peter Clark, Bhavana Dalvi Mishra, and Niket Tandon. 2018. What happened? leveraging verbnet to predict the effects of actions in procedural text. *arXiv preprint arXiv:1804.05435* .

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

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*. pages 1693–1701.

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

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proc. EMNLP'17*.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proc. ACL'17*.

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2008. A large-scale classification of english verbs. *Language Resources and Evaluation* 42(1):21–40.

Reginald Long, Panupong Pasupat, and Percy Liang. 2016. Simpler context-dependent logical forms via model projections. In *ACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pages 1532–1543.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proc. EMNLP'16*.

Karin Kipper Schuler. 2005. *VerbNet: A broad-coverage, comprehensive verb lexicon*. Ph.D. thesis, Univ Colorado at Boulder.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017a. Bidirectional attention flow for machine comprehension. In *Proc. ICLR'17*.

Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2017b. Query-reduction networks for question answering. In *ICLR*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2017. Constructing datasets for multi-hop reading comprehension across documents. *arXiv preprint arXiv:1710.06481* .

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* .

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916* .

Terry Winograd. 1972. Understanding natural language. *Cognitive Psychology* 3(1):1–191.

Matthew Zeiler. 2012. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .