# BUILDING DYNAMIC KNOWLEDGE GRAPHS FROM TEXT USING MACHINE READING COMPREHENSION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

We propose a neural machine-reading model that constructs dynamic knowledge graphs from procedural text. It builds these graphs recurrently for each step of the described procedure, and uses them to track the evolving states of participant entities. We harness and extend a recently proposed machine reading comprehension (MRC) model to query for entity states, since these states are generally communicated in spans of text and MRC models perform well in extracting entity-centric spans. The explicit, structured, and evolving knowledge graph representations that our model constructs can be used in downstream question answering tasks to improve machine comprehension of text, as we demonstrate empirically. On two comprehension tasks from the recently proposed PROPARA dataset, our model achieves state-of-the-art results. We further show that our model is competitive on the RECIPES dataset, suggesting it may be generally applicable. We present some evidence that the model's knowledge graphs help it to impose commonsense constraints on its predictions.

## 1 INTRODUCTION

Automatically building knowledge graphs (KGs) from text is a long-standing goal in artificial intelligence research. KGs organize raw information in a structured form, capturing relationships (labeled edges) between entities (nodes). They enable automated reasoning, e.g., the ability to infer unobserved facts from observed evidence and to make logical "hops," and render data amenable to decades of work in graph analysis.

There exists a profusion of text that describes complex, dynamic worlds in which entities' relationships evolve through time. This includes news articles, scientific manuals, and procedural text (e.g., recipes, how-to guides, and so on). Building KGs from this data would not only help us to study the changing relations among participant entities, but also to make implicit information more explicit. For example, the graphs at each step in Figure 1 help us to infer that the new entity *mixture* is created in the *leaf*, since the previous location of its participant entities (*light,* $CO_2$*, water*) was *leaf* – even though this is never stated in the text.

This paper introduces a neural machine-reading model, KG-MRC, that (i) explicitly constructs dynamic knowledge graphs to track state changes in procedural text and (ii) conditions on its own constructed knowledge graphs to improve downstream question answering on the text. Our dynamic graph model is recurrent, that is, the graph at each time step depends on the state of the graph at the previous time step. The constructed graphs are parameterized by real-valued embeddings for each node that change through time.

In text, entities and their states (e.g., their locations) are given by spans of words. Because of the variety of natural language, the same entity/state may be described with several surface forms. To address the challenge of entity/state recognition, our model uses a machine reading comprehension (MRC) mechanism (Seo et al., 2017a; Xiong et al., 2017; Chen et al., 2017; Yu et al., 2018, inter alia), which queries for entities and their states at each time step. We leverage MRC mechanisms because they have proven to be adept at extracting text spans that answer entity-centric questions (Levy et al., 2017). However, such models are static by design, returning the same answer for the same query and context. Since we expect answers about entity states to change over the course of the text, our model's MRC component conditions on the evolving graph at the current time step (this graph captures the instantaneous states of entities).

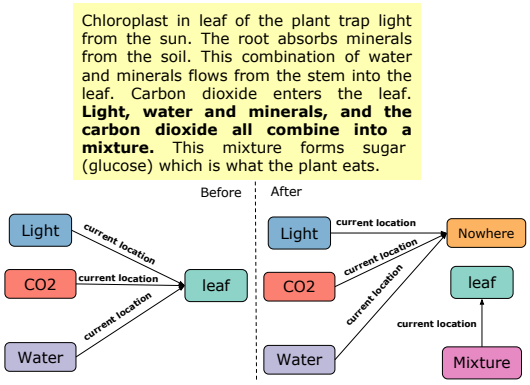


Figure 1: Snapshot of the knowledge graphs created by our model before and after reading the sentence in boldface. Since the KG explicitly stores the current location of *light*, $CO_2$, and *water* as *leaf*, the model can infer that *mixture* is formed in the *leaf* even though this is not explicitly stated. The three participant entities also get destroyed in the process, which is captured in the graph by pointing to a special *Nowhere* node.

To address the challenge of aliased text mentions, our model performs *soft co-reference* as it updates the graph. Instead of adding an alias node, like *the leaf* or *leaves* as aliases for *leaf*, the graph update procedure soft-attends (Bahdanau et al., 2014) over all nodes at the previous time step and performs a gated update (Cho et al., 2014; Chung et al., 2014) of the current embeddings with the previous ones. This ensures that state information is preserved and propagated across time steps. Soft co-reference can also handles the case that entity states *do not* change across time steps, by applying a near-null update to the existing state node rather than duplicating it.

At each time step, after the graph has been updated with the (possibly) new states of all entities, our model updates each entity representation with information about its state. The updated information about each individual entity is further propagated to all other entities (§ 4.4). This enables the model to recognize, for example, that entities are present in the same location (e.g., *light,* $CO_2$ and *water* in Figure 1). Thus, our model can use the information encoded in its internal knowledge graphs for a more comprehensive understanding of the text. We will demonstrate this experimentally by tackling comprehension tasks from the the recently released ProPara and Recipes datasets.

ProPara (Dalvi et al., 2018) consists of 488 human-authored paragraphs of procedural text, along with extensive annotation of state changes (location and existence of entities). It also introduces the task of tracking state changes at a fine-grained sentence level. A follow-up work (Tandon et al., 2018) introduces an additional task that evaluates state tracking at the process (paragraph) level. Both tasks and the dataset itself focus specifically on entities' location. Location is a state that can be tracked more reliably than others because it is usually stated at the surface level of the text.

Our complete machine reading model, which both builds and leverages dynamic knowledge graphs, can be trained end-to-end using only the loss from its MRC component; i.e., the negative log-likelihood that the MRC component assigns to the span that correctly describes each entity's queried state. We evaluate our model (KG-MRC) on the above two ProPara tasks and find that the same model significantly outperforms the previous state of the art. For example, KG-MRC obtains a *9.92%* relative improvement on the hard task of predicting at which time-step an entity moves. Similarly on the latter task, KG-MRC obtains a *5.7%* relative improvement over ProStruct and *41%* relative improvement over other entity-centric models such as EntNet (Henaff et al., 2017). On the related but much harder Recipes dataset, the same model obtains competitive performance.

## 2 Related Work

There are few datasets that address the challenging problem of tracking entity state changes. The bAbI dataset (Weston et al., 2015) includes questions about movement of entities; however, its language is generated synthetically over a small lexicon, and hence models trained on bAbI often do not generalize well when tested on real-world data. For example, state-of-the-art models like EntNet (Henaff et al., 2017) and Query Reduction Networks (Seo et al., 2017b) fail to perform well on ProPara.

ProRead (Berant et al., 2014) introduced the ProcessBank dataset, which contains paragraphs of procedural text as in ProPara. However, this earlier task involves mining arguments and relations from events, not tracking the dynamic state changes of entities. The model that Berant et al.

(2014) propose builds small knowledge graphs from the text, but they are not dynamic in nature. The model also relies on densely annotated process structure for training, demanding curation by domain experts. On the other hand, our model, KG-MRC, learns to build dynamic KGs just from annotations of text spans, which are much easier to collect.

For the sentence-level PROPARA task they propose, Dalvi et al. (2018) introduce two models: PROLOCAL and PROGLOBAL. PROLOCAL makes local predictions about entities by considering just the current sentence. This is followed by some heuristic/rule-based answer propagation. PROGLOBAL considers a broader context (previous sentences) and also includes the previous state of entities by considering the probability distribution over paragraph tokens in the previous step. Tandon et al. (2018) recently proposed a neural structured-prediction model, (PROSTRUCT), where hard and soft common-sense constraints are injected to steer their model away from globally incoherent predictions. We evaluate KG-MRC on the two PROPARA tasks proposed by Dalvi et al. (2018) and Tandon et al. (2018), respectively, and find that our single model outperforms each of the above models on their respective tasks of focus.

ENTNET (Henaff et al., 2017) and query reduction networks (QRN) (Seo et al., 2017b) are two state-of-the-art entity-centric models for the bAbI dataset. ENTNET maintains a dynamic memory of hidden states with a gated update to the memory slots at each step. Memory slots can be tied to specific entities, but unlike our model, ENTNET does not maintain separate embeddings of individual states (e.g., current locations); it also does not perform explicit co-reference updates. QRN refines the query vector as it processes each subsequent sentence until the query points to the answer, but does not maintain explicit representations of entity states. Neural Process Networks (NPN) (Bosselut et al., 2018) learn to understand procedural text by explicitly parameterizing actions and composing them with entities. These three models return an answer by predicting a vocabulary item in a multi-class classification setup, while in our work we predict spans of text directly from the paragraph.

MRC models have been used previously for extracting the argument of knowledge base (KB) relations, by associating one or more natural language questions with each relation (*querification*). These models have been shown to perform well in a zero-shot setting, i.e., for a previously unseen relation type (Levy et al., 2017), and for extracting entities that belong to non-standard types (Roth et al., 2018). These recent positive results motivate our use of an MRC component in KG-MRC.

## 3 DATA & TASKS

We evaluate KG-MRC on the recently released PROPARA dataset (Dalvi et al., 2018) containing procedural text about scientific processes. Annotators have labeled the location of entities in the process at each time step (sentence). The dataset also provides the names of the participant entities of the process. For example, for a process describing photosynthesis, the participant entities that are provided are: *light,* $CO_2$*, water, mixture* and *glucose*. Note that although the participant entities are given, the location of an entity could be *any* arbitrary span in the process text, making the task of finding an entity's location quite challenging. It should also be noted that the dataset does not provide information on whether a particular entity is an input to or output of a process. Not all entities exist from the beginning of the process (e.g. *glucose*) and not all exist at the end (e.g. *water*). Table 1 shows statistics of PROPARA. As can be seen, the training set is small, which makes learning challenging.

| | |
|---|---|
| # para | 488 |
| # train/#dev/#test | 391/43/54 |
| avg. # entities | 4.17 |
| avg. # sentences | 6.7 |
| # sentences | 3.3K |

Dalvi et al. (2018) introduce a task that measures the state change of entities at a fine-grained sentence level. To solve the task, a model must answer three categories of questions (10 questions in total) about an entity **e**: (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)? Cat. 1 asks boolean questions about the existence and movement of entities. Cat. 2 and 3 are harder tasks, as the model must correctly predict the step number at which a state changes as well as the correct locations (text spans) of entities at each step.

Tandon et al. (2018) introduced a new task on the same dataset that measures state changes at a coarser process level. To solve this task, a model must correctly answer the following four types

of questions: (1) What are the inputs to the process? (2) What are the outputs of the process? (3) What conversions occur, when and where? (4) What movements occur, when and where? Inputs to a process are defined as entities that exist at the start of the process but not at the end and outputs are entities that exist at the end of the process and were created during it. A conversion is when some entities are created and others destroyed, while movements refer to changes in location. Dalvi et al. (2018) and Tandon et al. (2018) introduce different models to solve each of these tasks, whereas we evaluate the same KG-MRC model on both tasks.

Bosselut et al. (2018) recently released the RECIPES dataset, which has various annotated states (e.g. shape, composition, location, etc.) for ingredients in cooking recipes. We tested our model on the location task to align with our PROPARA experiments. This is arguably the dataset's hardest task, since it requires classification over more than 260 classes while the others have a much smaller label space (maximum of 4). Note that rather than treating this problem as a classification over a fixed lexicon, our model aims to find the location-describing span of text in the recipe paragraph.

# 4 MODEL

KG-MRC is designed to track the temporal state change of entities in procedural text. Naturally, the model is entity-centric (Henaff et al., 2017; Bansal et al., 2017): it associates each participant entity of the procedural text (given in PROPARA *a priori*) with a unique encoding. KG-MRC is also equipped with a neural machine reading comprehension model which is queried about the current location of an entity.

Our model processes a paragraph $p = \{w_j\}_{j=1}^{P}$, consisting of $P$ words, by incrementally reading *prefixes* of the paragraph up to and including sentence $s_t$ for each time step $t$, until it has seen all sentences $\{s_t\}_{t=1}^{T}$ of that paragraph. For each time step, we construct a knowledge graph $G_t$ by modifying the graph $G_{t-1}$ from the previous time step.

The graph $G_t$ is modeled as bipartite, having two sets of nodes with implied connections between them: $G_t = \{e_{i,t}, \lambda_{i,t}\}$. Each node denotes either an entity $e_{i,t}$ or that entity's corresponding location $\lambda_{i,t}$, and is associated with a real-valued entity embedding updated across time steps. These bipartite graphs have only one (implicit) relation type, the current location, though we plan to extend this in future work. To derive $G_t$ from its previous iterate $G_{t-1}$, we combine both hard and soft graph updates. The update to an entity's node representation with new location information arises from a hard decision made by the MRC model, whereas co-reference between entities across time steps is resolved with soft attention. We now describe the various components of the model in detail.

## 4.1 ENTITY AND SPAN REPRESENTATIONS

In the PROPARA dataset, the participant entities of a process occur in the paragraph text.[1] Therefore, instead of using independent memory slots for each entity, we *derive* the entity representations from contextualized hidden vectors by encoding the paragraph text with a bi-directional LSTM (Hochreiter & Schmidhuber, 1997). This choice has the added advantage that the initial entity representations share information through context (Das et al., 2017; Bansal et al., 2017). Entities in the dataset can be multi-word expressions (e.g., *electric oven*). To obtain a single representation, we concatenate the contextualized hidden vectors corresponding to the start and end span tokens and take a linear projection. I.e., if the entity mention occurs between the $j$-th and $j+k$-th position, then the initial entity vector $v_i$ is computed as $v_i = W_e[c_j; c_{j+k}] + b_e$. We use $i$ to index an entity and its corresponding location, while $c_j$ represents the contextualized hidden vectors for token $j$ and $[;]$ represents the concatenate operation. An entity may occur multiple times within a paragraph. We give equal importance to all occurrences by summing the representations for each.

When queried about the current location of an entity, the MRC model (§ 4.2) returns a span of text as an answer, whose representation is later added to the appropriate node in the graph. We obtain this answer-span representation exactly as above.

[1] We compute the positions of the occurrence of entities by simple string matching.

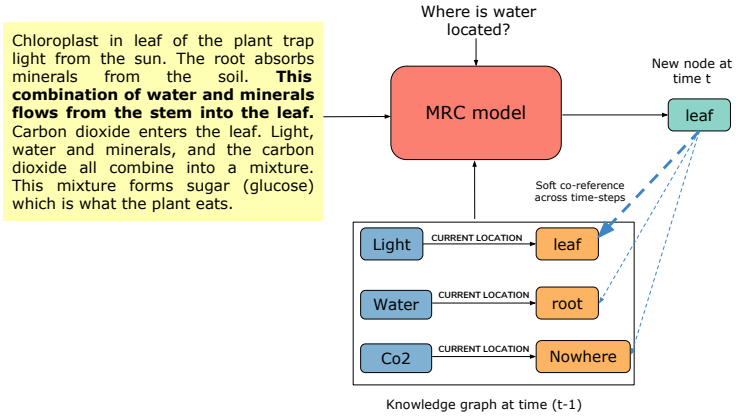


Figure 2: Soft co-reference across time steps. The sentence at the current time-step is highlighted. When the MRC model predicts a span (*leaf*) which is already present in the graph in the previous time step, KG-MRC does soft attention and a gated update to preserve the information across time steps (§ 4.3). The thicker arrow shows high attention weight between the old and the new node

### 4.2 MACHINE READING COMPREHENSION MODEL

KG-MRC is equipped with a machine reading comprehension model that it uses to query the state (current location) of an entity. Rather than design a specialized MRC architecture for our tasks, we make simple extensions to a widely used MRC model – DRQA (Chen et al., 2017) – that adapt it to query about the evolving states of entities. In summary, our modified DRQA implementation operates on prefixes of sentences rather than the full paragraph (like PROGLOBAL), and at each sentence (time step) it conditions on both the current sentence representation $s_t$ and dynamic entity representations in $G_t$.

For complete details of the DRQA model, we refer readers to the original publication (Chen et al., 2017). Broadly, it uses a multi-layer recurrent neural network (RNN) architecture for encoding both the passage and question text and uses self-attention to match these two encodings. For each token $j$ in the text, it outputs a score indicating its likelihood of being the start or end of the span that answers the question. We reuse all of these operations in our model, modified as described below.

We query the DRQA model about the state of each participant entity at each time step $t$, which involves reading the paragraph up to and including sentence $s_t$. To query, we generate simple natural language questions for each entity, $e$, such as "Where is $e$ located?" This is motivated by the work of Levy et al. (2017). Our DRQA component also conditions on the entity that is being queried about. Let $e$ denote the current representation of entity $e$ in the full model's graphical representation. The DRQA component conditions on $e$ in its output layer, similar to the way the question representation is used in the *output alignment* step in Chen et al. (2017). However, instead of taking a bi-linear map between $p_i$ and $q$ as in that work, we first concatenate the question representation with the entity representation $e$ and then pass the concatenation through a 2-layer MLP, obtaining an entity-dependent question representation. We use this to compute the output start and end scores for each token position, taking the arg max to obtain the most likely span and adding this to the graph.

### 4.3 SOFT CO-REFERENCE

To handle cases when entity states do not change and when states are referred to with different surface forms (either of which could lead to undesired node duplication), our model uses soft co-reference mechanisms both across and within time steps. Disambiguation across time steps is accomplished by attention and a gated update as follows:

$$
\begin{aligned}
a_{i,t} &= \text{softmax}(\Lambda_{t-1}^{\top} v_i) \\
v'_{i,t} &= \Lambda_{t-1} a_{i,t} \\
g_{i,t} &= \text{sigmoid}(W_i[v'_{i,t}; v_i] + b_i) \\
\lambda'_{i,t} &= g_{i,t} v_{i,t} + (1 - g_{i,t}) v'_i,
\end{aligned}
\tag{1}
$$

where $\Lambda_{t-1} = \{\lambda^L_{i,t-1}\}_{i=1}^N$ is a set of location node representations from the previous time step and $v_i$ is the initial entity vector. The result vector $\lambda'_{i,t}$ is a disambiguated intermediate node representation. For the first time step and when the graph $G_0$ is empty, this intermediate representation is set to the initial entity vector, i.e., $\lambda'_{i,0} = v_i$. This process only partially addresses node de-duplication. Since different instances of the same location can be predicted for multiple entities, we also perform a co-reference disambiguation within each time step with a self-attention mechanism:

$$\begin{aligned} u_{i,t} &= \text{softmax}(\Lambda'^{\top}_t \lambda'_{i,t}) \\ \lambda_{i,t} &= \Lambda'_t u_{i,t}, \end{aligned} \tag{2}$$

where $\Lambda'_t = \{\lambda'_{i,t}\}_{i=1}^N$ is the set of intermediate representations of the nodes and $U_t = \{u_{i,t}\}_{i=1}^N$ is a co-reference adjacency matrix. We calculate this adjacency matrix $U_t$ at the beginning of each time step $t$ so that it may be used it to track related nodes within that time step.

### 4.4 Graph Update

Our graph module updates and tracks the states of all entity and location nodes through both space and time. The graph update proceeds as follows. We first compose all connected entity and location nodes with an LSTM unit: $h^l_{i,t} = \text{LSTM}([e^{l-1}_{i,t}; \lambda^{l-1}_{i,t}])$. Next, node-specific information is attached to the entity representations with a residual update (He et al., 2016):

$$\begin{aligned} e^l_{i,t} &= e^{l-1}_{i,t} + h^l_{i,t} \\ \lambda'^l_{i,t} &= \lambda^{l-1}_{i,t} + h^l_{i,t}. \end{aligned} \tag{3}$$

For location-specific nodes, we perform co-reference pooling as above in each layer $l$, using the co-reference adjacency matrix: $\lambda^l_{i,t} = \Lambda'^l_t u_{i,t}$, where $\Lambda'^l_t = \{\lambda'^l_{i,t}\}_{i=1}^N$.

The recurrent graph module stacks $L$ such layers to propagate node information along the graph's edges. The resulting node representations $e^L_{i,t}$ and $\lambda^L_{i,t}$ for each participant entity and its location are used to condition the MRC model as described in §4.2. We make use of this particular graph module structure, rather than adopting an existing model like GraphCNNs (Edwards & Xie, 2016), because recurrent networks are designed to propagate information through time.

### 4.5 Training

The full KG-MRC model is trained end-to-end by minimizing the negative log-likelihood of the correct span tokens under the MRC module's output distribution. This is a fairly soft supervision signal, since we do not train the graph construction modules directly. We teacher-force the model at training time by updating the location-node representations with the encoding of the correct span. We do not pretrain the MRC module, but we represent paragraph tokens with pretrained FastText embeddings (Joulin et al., 2016). See the appendix A for full implementation and training details.

## 5 Experiments and Discussion

We evaluate our model on three different tasks. We also provide an ablation study along with quantitative and qualitative analyses to highlight the performance contributions of each module.

### 5.1 Results on Procedural Text

We benchmarked our model on two ProPara comprehension tasks introduced respectively in Dalvi et al. (2018) and Tandon et al. (2018). Both tasks are entity-centric and require a model to reason from text about temporal state changes. Refer to Section 3 for a detailed description about the data and tasks. In results below, we report an average of 3 runs of our model with random seeds.

#### 5.1.1 Task 1: Sentence-level Evaluation

Table 1 shows our main results on the first task. Following the original task evaluation[2], we report model accuracy on each subtask category and macro and micro averages over the subtasks.

[2]https://github.com/allenai/propara/blob/master/propara/eval/evalQA.py

Human performance is 79.69%, micro-average. A state-of-the-art memory augmented network, ENTNET (Henaff et al., 2017), which is built to track entities but lacks an explicit graph structure, achieves 25.96%. The previous best performing model is PROGLOBAL, which achieves 45.37%. Our KG-MRC improves over this result by 1.25% absolute score in terms of micro-averaged accuracy. Comparing various models for each subtask category, PROGLOBAL leads in Category 1 by a small margin of around 0.1%. For the more challenging Categories 2 and 3, KG-MRC outperforms PROGLOBAL by a large margin. These questions require a model to make fine-grained predictions of state changes.

| | Cat 1 | Cat 2 | Cat 3 | Macro-avg | Micro-avg |
|---|---|---|---|---|---|
| Human upper bound | 91.67 | 87.66 | 62.96 | 80.76 | 79.69 |
| Majority | 51.01 | – | – | – | – |
| Rule based | 57.14 | 20.33 | 2.40 | 26.62 | 26.24 |
| Feature based | 58.64 | 20.82 | 9.66 | 29.7 | 29.64 |
| EntNet (Henaff et al. (2017)) | 51.62 | 18.83 | 7.77 | 26.07 | 25.96 |
| Pro-Local (Dalvi et al. (2018)) | 62.65 | 30.50 | 10.35 | 34.50 | 33.96 |
| Pro-Global (Dalvi et al. (2018)) | **62.95** | 36.39 | 35.90 | 45.08 | 45.37 |
| KG-MRC (ours) | 62.86 | **40.00** | **38.23** | **47.03** | **46.62** |

Table 1: Task 1 results (accuracy).

#### 5.1.2 TASK 2: DOCUMENT-LEVEL EVALUATION

We report the performance of our model on the document-level task, along with previously published results, in Table 2. The same KG-MRC model achieves 3.02% absolute improvement in F1 over the previous best result of PROSTRUCT. PROSTRUCT incorporates a set of commonsense constraints for globally consistent predictions. We analyzed KG-MRC's outputs and were surprised to discover that our model learns these commonsense constraints from the data in an end-to-end fashion, as we show quantitatively in §5.4.

| | Precision | Recall | F1 |
|---|---|---|---|
| Pro-Local (Dalvi et al. (2018)) | **77.4** | 22.9 | 35.3 |
| QRN (Seo et al. (2017b)) | 55.5 | 31.3 | 40.0 |
| EntNet (Henaff et al. (2017)) | 50.2 | 33.5 | 40.2 |
| Pro-Global (Dalvi et al. (2018)) | 46.7 | **52.4** | 49.4 |
| Pro-Struct (Tandon et al. (2018)) | 74.2 | 42.1 | 53.75 |
| KG-MRC (ours) | 64.52 | 50.68 | **56.77** |

Table 2: Task 2 results.

### 5.2 RECIPE DESCRIPTION EXPERIMENTS

We also evaluate our model on the RECIPES dataset, where we predict the location of cooking ingredients during procedural food preparation. In the original work of Bosselut et al. (2018), they treat this problem as multi-class classification over a fixed lexicon of locations, whereas KG-MRC searches for the correct location span in the text. Our model slightly outperforms the baseline NPN model on this task even after it was trained on just 10K examples (the full training set is around 60K examples): NPN achieves 51.28% F1 training on all the data, while KG-MRC achieves 51.64% F1 after 10k training samples.

### 5.3 ABLATION STUDY

We performed an ablation study to evaluate different model variations on PROPARA Task 1. The main results are reported in Table 3. Removing the soft co-reference disambiguation within time steps (Equations 2) from KG-MRC resulted in around 1% performance drop. The drop is more significant when the co-reference disambiguation across time steps (Equations 1) is removed.

We also replaced the recurrent graph module with the standard LSTM unit and used the LSTM hidden state for the entity representation. As this model variation lacks the information propagation across graph nodes, we observed a large performance decrease.

For the last two variations, we simply train the MRC model in isolation and predict location spans from the current sentence or paragraph prefix text (i.e., the current and all previous sentences). These models construct no internal knowledge graphs. We can see that training the MRC model on paragraph prefixes already provides a good starting performance of 40.83% micro-average, which is significantly boosted by the recurrent graph module and graph conditioning up to 47.64%.

| | Cat 1 | Cat 2 | Cat 3 | Macro-avg | Micro-avg |
|---|---|---|---|---|---|
| KG-MRC | 58.55 | 38.52 | 42.22 | 46.43 | 47.64 |
| - Coref across time steps | 61.07 | 37.38 | 35.58 | 44.68 | 46.32 |
| - Coref within time step | 57.88 | 38.09 | 40.19 | 45.39 | 46.63 |
| Standard LSTM as graph unit | 56.84 | 13.15 | 10.95 | 26.98 | 29.97 |
| MRC on current sentence | 58.85 | 21.82 | 26.52 | 35.73 | 35.98 |
| MRC on prefix | 61.28 | 32.58 | 29.48 | 41.11 | 40.83 |

Table 3: Ablation experiment results

### 5.4 COMMONSENSE CONSTRAINTS

For accurate globally consistent predictions for the second task on procedural text, (Tandon et al., 2018) introduced a set of commonsense constraints as follows: 1) An entity must **exist** before it can be **moved** or **destroyed**; 2) An entity cannot be **created** if it already **exists**; 3) An entity cannot **change** until it is **mentioned** in the paragraph. To quantitatively analyze whether our model learns the above constraints from the data, we count the number of predictions that violate any of these constraints using the test set. To our surprise, KG-MRC produces 0 constraint violations across the 3 categories. This is learned purely from the data. The model seems to capture these commonsense constraints via its dynamic graph component.

### 5.5 QUALITATIVE ANALYSIS

We picked an example from the test data and had a closer look at the model outputs to investigate how KG-MRC dynamically adjusts its decisions via the dynamic graph module and finds accurate spans with the conditional MRC model. The step-by-step output of both PROGLOBAL (Dalvi et al. (2018)) and KG-MRC is shown in Table 4, where we track the state of entity *blood* across six sentences. KG-MRC outputs smoother and more accurate predictions.

| Sentences | Location of entities after each sentence | |
|---|---|---|
| (Before first sentence) | somewhere | somewhere |
| Blood enters the right side of your heart. | heart | right side of your heart |
| Blood travels to the lungs. | lung | lungs |
| Carbon dioxide is removed from the blood. | blood | lungs |
| Oxygen is added to your blood. | lung | lungs |
| Blood returns to left side of your heart. | blood | heart |
| The blood travels through the body. | body | body |

Table 4: Two models' predictions of entity locations, on randomly selected paragraph about blood circulation. In this example the entity is **blood**. Predicted results from Pro-Local (Dalvi et al. (2018)) are in orange, results from KG-MRC are in red, important locations in paragraph are in blue.

## 6 CONCLUSION

We proposed a neural machine-reading model that constructs dynamic knowledge graphs from text to track locations of participant entities in procedural text. It further uses these graphical representations to improve its downstream comprehension of text. Our model, KG-MRC, achieves state-of-the-art results on two question-answering tasks from the PROPARA dataset and one from the RECIPES dataset. We present some evidence that the knowledge graphs built by the model help it to impose commonsense constraints on its predictions. In future work, we will extend the model to construct more general knowledge graphs with multiple relation types.

## REFERENCES


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Trapit Bansal, Arvind Neelakantan, and Andrew McCallum. Relnet: End-to-end modeling of entities & relations. In *AKBC, NIPS*, 2017.

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

Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. Simulating action dynamics with neural process networks. In *ICLR*, 2018.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. In *ACL*, 2017.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

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

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

Rajarshi Das, Manzil Zaheer, Siva Reddy, and Andrew McCallum. Question answering on knowledge bases and text using universal schema and memory networks. In *ACL*, 2017.

Michael Edwards and Xianghua Xie. Graph based convolutional neural network. *arXiv preprint arXiv:1609.08965*, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. Tracking the world state with recurrent entity networks. In *ICLR*, 2017.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016.

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

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke S. Zettlemoyer. Zero-shot relation extraction via reading comprehension. In *CoNLL*, 2017.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

Benjamin Roth, Costanza Conforti, Nina Poerner, Sanjeev Karn, and Hinrich Schütze. Neural architectures for open-type relation argument extraction. *arXiv preprint arXiv:1803.01707*, 2018.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. In *ICLR*, 2017a.

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

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

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

Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. In *ICLR*, 2017.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. Qanet: Combining local convolution with global self-attention for reading comprehension. In *ICLR*, 2018.

# A IMPLEMENTATION DETAILS

Implementation details of KG-MRC are as follows.
In all experiments, the word embeddings are initialized with FastText embeddings Joulin et al. (2016); we use a document LSTM with two layers, the number of hidden units in each layer is 64. We apply dropout rate of 0.4 in all recurrent layers, and 0.3 in all other layers.

During training, the mini-batch size is 8. We use *adam* (Kingma & Ba, 2014) as the step rule for optimization, The learning rate is set to 0.002. The model is implemented using *PyTorch* (Paszke et al., 2017).