Question Answering by Reasoning Across Documents with Graph Convolutional Networks

Citation for published version:

Digital Object Identifier (DOI):
10.18653/v1/n19-1240

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
Abstract

Most research in reading comprehension has focused on answering questions based on individual documents or even single paragraphs. We introduce a neural model which integrates and reasons relying on information spread within documents and across multiple documents. We frame it as an inference problem on a graph. Mentions of entities are nodes of this graph while edges encode relations between different mentions (e.g., within- and cross-document coreference). Graph convolutional networks (GCNs) are applied to these graphs and trained to perform multi-step reasoning. Our Entity-GCN method is scalable and compact, and it achieves state-of-the-art results on a multi-document question answering dataset, WIKIHOP (Welbl et al., 2018).

1 Introduction

The long-standing goal of natural language understanding is the development of systems which can acquire knowledge from text collections. Fresh interest in reading comprehension tasks was sparked by the availability of large-scale datasets, such as SQuAD (Rajpurkar et al., 2016) and CNN/Daily Mail (Hermann et al., 2015), enabling end-to-end training of neural models (Seo et al., 2016; Xiong et al., 2016; Shen et al., 2017). These systems, given a text and a question, need to answer the query relying on the given document.

Recently, it has been observed that most questions in these datasets do not require reasoning across the document, but they can be answered relying on information contained in a single sentence (Weissenborn et al., 2017). The last generation of large-scale reading comprehension datasets, such as a NarrativeQA (Kocisky et al., 2018), TriviaQA (Joshi et al., 2017), and RACE (Lai et al., 2017), have been created in such a way as to address this shortcoming and to ensure that systems relying only on local information cannot achieve competitive performance.

Even though these new datasets are challenging and require reasoning within documents, many question answering and search applications require aggregation of information across multiple documents. The WIKIHOP dataset (Welbl et al., 2018) was explicitly created to facilitate the development of systems dealing with these scenarios. Each example in WIKIHOP consists of a collection of documents, a query and a set of candidate answers (Figure 1). Though there is no guarantee that a question cannot be answered by relying just on a single sentence, the authors ensure that it is answerable using a chain of reasoning crossing document boundaries.

Though an important practical problem, the multi-hop setting has so far received little attention. The methods reported by Welbl et al. (2018) approach the task by merely concatenating all documents into a single long text and training a standard RNN-based reading comprehension model, namely, BiDAF (Seo et al., 2016) and FastQA (Weissenborn et al., 2017). Document concatenation in this setting is also used in Weaver (Raison et al., 2018) and MHPGM (Bauer et al., 2018). The only published paper which
goes beyond concatenation is due to Dhingra et al. (2018), where they augment RNNs with jump-links corresponding to co-reference edges. Though these edges provide a structural bias, the RNN states are still tasked with passing the information across the document and performing multi-hop reasoning.

Instead, we frame question answering as an inference problem on a graph representing the document collection. Nodes in this graph correspond to named entities in a document whereas edges encode relations between them (e.g., cross- and within-document coreference links or simply co-occurrence in a document). We assume that reasoning chains can be captured by propagating local contextual information along edges in this graph using a graph convolutional network (GCN) (Kipf and Welling, 2017).

The multi-document setting imposes scalability challenges. In realistic scenarios, a system needs to learn to answer a query for a given collection (e.g., Wikipedia or a domain-specific set of documents). In such scenarios one cannot afford to run expensive document encoders (e.g., RNN or transformer-like self-attention (Vaswani et al., 2017)), unless the computation can be preprocessed both at train and test time. Even if (similarly to WikIHop creators) one considers a coarse-to-fine approach, where a set of potentially relevant documents is provided, re-encoding them in a query-specific way remains the bottleneck. In contrast to other proposed methods (e.g., (Dhingra et al., 2018; Raison et al., 2018; Seo et al., 2016)), we avoid training expensive document encoders.

In our approach, only a small query encoder, the GCN layers and a simple feed-forward answer selection component are learned. Instead of training RNN encoders, we use contextualized embeddings (ELMo) to obtain initial (local) representations of nodes. This implies that only a lightweight computation has to be performed online, both at train and test time, whereas the rest is preprocessed. Even in the somewhat contrived WikIHop setting, where fairly small sets of candidates are provided, the model is at least 5 times faster to train than BiDAF. 1 Interestingly, when we substitute ELMo with simple pre-trained word embeddings, Entity-GCN still performs on par with many techniques that use expensive question-aware recurrent document encoders.

Despite not using recurrent document encoders, the full Entity-GCN model achieves over 2% improvement over the best previously-published results. As our model is efficient, we also reported results of an ensemble which brings further 3.6% of improvement and only 3% below the human performance reported by Welbl et al. (2018). Our contributions can be summarized as follows:

- we present a novel approach for multi-hop QA that relies on a (pre-trained) document encoder and information propagation across multiple documents using graph neural networks;
- we provide an efficient training technique which relies on a slower offline and a faster on-line computation that does not require expensive document processing;
- we empirically show that our algorithm is effective, presenting an improvement over previous results.

2 Method

In this section we explain our method. We first introduce the dataset we focus on, WikIHop by Welbl et al. (2018), as well as the task abstraction. We then present the building blocks that make up our Entity-GCN model, namely, an entity graph used to relate mentions to entities within and across documents, a document encoder used to obtain representations of mentions in context, and a relational graph convolutional network that propagates information through the entity graph.

2.1 Dataset and task abstraction

Data The WikIHop dataset comprises of tuples $\langle q, S_q, C_q, a^{*}\rangle$ where: $q$ is a query/question, $S_q$ is a set of supporting documents, $C_q$ is a set of candidate answers (all of which are entities mentioned in $S_q$), and $a^{*} \in C_q$ is the entity that correctly answers the question. WikIHop is assembled assuming that there exists a corpus and a knowledge base (KB) related to each other. The KB contains triples $\langle s, r, o \rangle$ where $s$ is a subject entity, $o$ an object entity, and $r$ a unidirectional relation between them. Welbl et al. (2018) used WIKIPEDIA as corpus and WIKIDATA (Vrandečić, 2012) as KB. The KB is only used for constructing WikIHop: Welbl
et al. (2018) retrieved the supporting documents $S_q$ from the corpus looking at mentions of subject and object entities in the text. Note that the set $S_q$ (not the KB) is provided to the QA system, and not all of the supporting documents are relevant for the query but some of them act as distractors. Queries, on the other hand, are not expressed in natural language, but instead consist of tuples $(s, r, ?)$ where the object entity is unknown and it has to be inferred by reading the support documents. Therefore, answering a query corresponds to finding the entity $a^*$ that is the object of a tuple in the KB with subject $s$ and relation $r$ among the provided set of candidate answers $C_q$.

**Task** The goal is to learn a model that can identify the correct answer $a^*$ from the set of supporting documents $S_q$. To that end, we exploit the available supervision to train a neural network that computes scores for candidates in $C_q$. We estimate the parameters of the architecture by maximizing the likelihood of observations. For prediction, we then output the candidate that achieves the highest probability. In the following, we present our model discussing the design decisions that enable multi-step reasoning and an efficient computation.

### 2.2 Reasoning on an entity graph

**Entity graph** In an offline step, we organize the content of each training instance in a graph connecting mentions of candidate answers within and across supporting documents. For a given query $q = (s, r, ?)$, we identify mentions in $S_q$ of the entities in $C_q \cup \{s\}$ and create one node per mention. This process is based on the following heuristic:

1. we consider mentions spans in $S_q$ exactly matching an element of $C_q \cup \{s\}$. Admittedly, this is a rather simple strategy which may suffer from low recall.

2. we use predictions from a coreference resolution system to add mentions of elements in $C_q \cup \{s\}$ beyond exact matching (including both noun phrases and anaphoric pronouns). In particular, we use the end-to-end coreference resolution by Lee et al. (2017).

3. we discard mentions which are ambiguously resolved to multiple coreference chains; this may sacrifice recall, but avoids propagating ambiguity.

To each node $v_i$, we associate a continuous annotation $x_i \in \mathbb{R}^D$ which represents an entity in the context where it was mentioned (details in Section 2.3). We then proceed to connect these mentions i) if they co-occur within the same document (we will refer to this as **DOC-BASED** edges), ii) if the pair of named entity mentions is identical (**MATCH** edges—these may connect nodes across and within documents), or iii) if they are in the same coreference chain, as predicted by the external coreference system (**COREF** edges). Note that **MATCH** edges when connecting mentions in the same document are mostly included in the set of edges predicted by the coreference system. Having the two types of edges lets us distinguish between less reliable edges provided by the coreference system and more reliable (but also more sparse) edges given by the exact-match heuristic. We treat these three types of connections as three different types of relations. See Figure 2 for an illustration. In addition to that, and to prevent having disconnected graphs, we add a fourth type of relation (**COMPLEMENT** edge) between any two nodes that are not connected with any of the other relations. We can think of these edges as those in the complement set of the entity graph with respect to a fully connected graph.

**Multi-step reasoning** Our model then approaches multi-step reasoning by transforming node representations (Section 2.3 for details) with a differentiable message passing algorithm that propagates information through the entity graph.
The algorithm is parametrized by a graph convolutional network (GCN) (Kipf and Welling, 2017), in particular, we employ relational-GCNs (Schlichtkrull et al., 2018), an extended version that accommodates edges of different types. In Section 2.4 we describe the propagation rule.

Each step of the algorithm (also referred to as a hop) updates all node representations in parallel. In particular, a node is updated as a function of messages from its direct neighbours, and a message is possibly specific to a certain relation. At the end of the first step, every node is aware of every other node it connects directly to. Besides, the neighbourhood of a node may include mentions of the same entity as well as others (e.g., same-document relation), and these mentions may have occurred in different documents. Taking this idea recursively, each further step of the algorithm allows a node to indirectly interact with nodes already known to their neighbours. After \( L \) layers of R-GCN, information has been propagated through paths connecting up to \( L + 1 \) nodes.

We start with node representations \( \{ \mathbf{h}_i^{(0)} \}_{i=1}^N \), and transform them by applying \( L \) layers of R-GCN obtaining \( \{ \mathbf{h}_i^{(L)} \}_{i=1}^N \). Together with a representation \( \mathbf{q} \) of the query, we define a distribution over candidate answers and we train maximizing the likelihood of observations. The probability of selecting a candidate \( c \in C_q \) as an answer is then

\[
P(c|q, C_q, S_q) \propto \exp \left( \max_{i \in \mathcal{M}_c} f_o([\mathbf{q}, \mathbf{h}_i^{(L)}]) \right),
\]

where \( f_o \) is a parametrized affine transformation, and \( \mathcal{M}_c \) is the set of node indices such that \( i \in \mathcal{M}_c \) only if node \( v_i \) is a mention of \( c \). The \( \max \) operator in Equation 1 is necessary to select the node with highest predicted probability since a candidate answer is realized in multiple locations via different nodes.

2.3 Node annotations

Keeping in mind we want an efficient model, we encode words in supporting documents and in the query using only a pre-trained model for contextualized word representations rather than training our own encoder. Specifically, we use ELMo\(^2\) (Peters et al., 2018), a pre-trained bi-directional language model that relies on character-based input representation. ELMo representations, differently from other pre-trained word-based models (e.g., word2vec (Mikolov et al., 2013) or GloVe (Pennington et al., 2014)), are contextualized since each token representation depends on the entire text excerpt (i.e., the whole sentence).

We choose not to fine tune nor propagate gradients through the ELMo architecture, as it would have defied the goal of not having specialized RNN encoders. In the experiments, we will also ablate the use of ELMo showing how our model behaves using non-contextualized word representations (we use GloVe).

Documents pre-processing ELMo encodings are used to produce a set of representations \( \{ \mathbf{x}_i \}_{i=1}^N \), where \( \mathbf{x}_i \in \mathbb{R}^D \) denotes the \( i \)th candidate mention in context. Note that these representations do not depend on the query yet and no trainable model was used to process the documents so far, that is, we use ELMo as a fixed pre-trained encoder. Therefore, we can pre-compute representation of mentions once and store them for later use.

Query-dependent mention encodings ELMo encodings are used to produce a query representation \( \mathbf{q} \in \mathbb{R}^K \) as well. Here, \( \mathbf{q} \) is a concatenation of the final outputs from a bidirectional RNN layer trained to re-encode ELMo representations of words in the query. The vector \( \mathbf{q} \) is used to compute a query-dependent representation of mentions \( \{ \hat{\mathbf{x}}_i \}_{i=1}^N \) as well as to compute a probability distribution over candidates (as in Equation 1). Query-dependent mention encodings \( \hat{\mathbf{x}}_i = f_x(\mathbf{q}, \mathbf{x}_i) \) are generated by a trainable function \( f_x \) which is parameterized by a feed-forward neural network.

2.4 Entity relational graph convolutional network

Our model uses a gated version of the original R-GCN propagation rule. At the first layer, all hidden node representation are initialized with the query-aware encodings \( \mathbf{h}_i^{(0)} = \hat{\mathbf{x}}_i \). Then, at each layer \( 0 \leq \ell \leq L \), the update message \( \mathbf{u}_i^{(\ell)} \) to the \( i \)th node is a sum of a transformation \( f_s \) of the current node representation \( \mathbf{h}_i^{(\ell)} \) and transformations of its neighbours:

\[
\mathbf{u}_i^{(\ell)} = f_s(\mathbf{h}_i^{(\ell)}) + \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \sum_{r \in \mathcal{R}_{ij}} f_r(\mathbf{h}_j^{(\ell)}),
\]

where \( |\mathcal{N}_i| \) is the set of indices of nodes neighbouring the \( i \)th node, \( \mathcal{R}_{ij} \) is the set of edge annotations.
between \( i \) and \( j \), and \( f_r \) is a parametrized function specific to an edge type \( r \in \mathcal{R} \). Recall the available relations from Section 2.2, namely, \( \mathcal{R} = \{ \text{DOC-BASED}, \text{MATCH}, \text{COREF}, \text{COMPLEMENT} \} \).

A gating mechanism regulates how much of the update message propagates to the next step. This provides the model a way to prevent completely overwriting past information. Indeed, if all necessary information to answer a question is present at a layer which is not the last, then the model should learn to stop using neighbouring information for the next steps. Gate levels are computed as

\[
a_i^{(\ell)} = \sigma \left( f_a \left( [u_i^{(\ell)}, h_i^{(\ell)}] \right) \right),
\]

where \( \sigma(\cdot) \) is the sigmoid function and \( f_a \) a parametrized transformation. Ultimately, the updated representation is a gated combination of the previous representation and a non-linear transformation of the update message:

\[
h_i^{(\ell+1)} = \phi(u_i^{(\ell)}) \odot a_i^{(\ell)} + h_i^{(\ell)} \odot (1 - a_i^{(\ell)}),
\]

where \( \phi(\cdot) \) is any nonlinear function (we used tanh) and \( \odot \) stands for element-wise multiplication. All transformations \( f_s \) are affine and they are not layer-dependent (since we would like to use as few parameters as possible to decrease model complexity promoting efficiency and scalability).

## 3 Experiments

In this section, we compare our method against recent work as well as preforming an ablation study using the WIKIHOP dataset (Welbl et al., 2018). See Appendix A in the supplementary material for a description of the hyper-parameters of our model and training details.

**WIKIHOP** We use WIKIHOP for training, validation/development and test. The test set is not publicly available and therefore we measure performance on the validation set in almost all experiments. WIKIHOP comes in two versions, a standard (unmasked) one and a masked one. The masked version was created by the authors to test whether methods are able to learn lexical abstraction. In this version, all candidates and all mentions of them in the support documents are replaced by random but consistent placeholder tokens. Thus, in the masked version, mentions are always referred to via unambiguous surface forms. We do not use coreference systems in the masked version as they rely crucially on lexical realization of mentions and cannot operate on masked tokens.

### 3.1 Comparison

In this experiment, we compare our Entity-GCN against recent prior work on the same task. We present test and development results (when present) for both versions of the dataset in Table 1. From Welbl et al. (2018), we list an oracle based on human performance as well as two standard reading comprehension models, namely BiDAF (Seo et al., 2016) and FastQA (Weissenborn et al., 2017). We also compare against Coref-GRU (Dhingra et al., 2018), MHPGM (Bauer et al., 2018), and Weaver (Raison et al., 2018). Additionally, we include results of MHQA-GRN (Song et al., 2018), from a recent arXiv preprint describing concurrent work. They jointly train graph neural networks and recurrent encoders. We report single runs of our best models on the unmasked test set (recall that the test set is not publicly available and the task organizers only report unmasked results) as well as both versions of the validation set.

Entity-GCN (best single model without coreference edges) outperforms all previous work by over 2\% points. We also report an ensemble of 5 independently trained models. We additionally re-ran BiDAF baseline to compare training time: when using a single Titan X GPU, BiDAF and Entity-GCN process 12.5 and 57.8 document sets per second, respectively. Note that Welbl et al. (2018) had to use BiDAF with very small state dimensionalities (20), and smaller batch size due to the scalability issues (both memory and computation costs). We compare applying the same reductions.\(^3\)

### 3.2 Ablation study

To help determine the sources of improvements, we perform an ablation study using the publicly available validation set (see Table 2). We perform two groups of ablation, one on the embedding layer, to study the effect of ELMo, and one on the edges, to study how different relations affect the overall model performance.

**Embedding ablation** We argue that ELMo is crucial, since we do not rely on any other context encoder. However, it is interesting to explore how our R-GCN performs without it. Therefore, in this experiment, we replace the deep contextualized embeddings of both the query and the nodes with

\(^3\)Besides, we could not run any other method we compare with combined with ELMo without reducing the dimensionality further or having to implement a distributed version.
Table 1: Accuracy of different models on Wikihop closed test set and public validation set. Our Entity-GCN outperforms recent prior work without learning any language model to process the input but relying on a pre-trained one (ELMo – without fine-tuning it) and applying R-GCN to reason among entities in the text.

GloVe (Pennington et al., 2014) vectors (insensitive to context). Since we do not have any component in our model that processes the documents, we expect a drop in performance. In other words, in this ablation our model tries to answer questions without reading the context at all. For example, in Figure 1, our model would be aware that “Stockholm” and “Sweden” appear in the same document but any context words, including the ones encoding relations (e.g., “is the capital of”) will be hidden. Besides, in the masked case all mentions become ‘unknown’ tokens with GloVe and therefore the predictions are equivalent to a random guess. Once the strong pre-trained encoder is out of the way, we also ablate the use of our R-GCN component, thus completely depriving the model from inductive biases that aim at multi-hop reasoning.

The first important observation is that replacing ELMo by GloVe (GloVe with R-GCN in Table 2) still yields a competitive system that ranks far above baselines from (Welbl et al., 2018) and even above the Coref-GRU of Dhingra et al. (2018), in terms of accuracy on (unmasked) validation set. The second important observation is that if we then remove R-GCN (GloVe w/o R-GCN in Table 2), we lose 8.0 points. That is, the R-GCN component pushes the model to perform above Coref-GRU still without accessing context, but rather by updating mention representations based on their relation to other ones. These results highlight the impact of our R-GCN component.

Graph edges ablation In this experiment we investigate the effect of the different relations available in the entity graph and processed by the R-GCN module. We start off by testing our stronger encoder (i.e., ELMo) in absence of edges connecting mentions in the supporting documents (i.e., using only self-loops – No R-GCN in Table 2). The results suggest that Wikihop genuinely requires multihop inference, as our best model is 6.1% and 8.4% more accurate than this local model, in unmasked and masked settings, respectively.4 However, it also shows that ELMo representations capture predictive context features, without being explicitly trained for the task. It confirms that our goal of getting away with training expensive document encoders is a realistic one.

We then inspect our model’s effectiveness in making use of the structure encoded in the graph. We start naively by fully-connecting all nodes within and across documents without distinguishing edges by type (No relation types in Table 2). We observe only marginal improvements with respect to ELMo alone (No R-GCN in Table 2) in both the unmasked and masked setting suggesting that a GCN operating over a naive entity graph would not add much to this task and a more informative graph construction and/or a more sophisticated parameterization is indeed needed.

Next, we ablate each type of relations independently, that is, we either remove connections of mentions that co-occur in the same document (DOC-BASED), connections between mentions matching exactly (MATCH), or edges pre-

4Recall that all models in the ensemble use the same local representations, ELMo.
The last row of Table 2 (Induced edges) shows labels by a model component that predicts them. place our heuristic for assigning edges and their role. Though it may be surprising for coreference connections even though we do not apply the coreference system in this setting due to masking. It is not surprising as coreferred mentions are labeled with the same identifier in the masked version, even if their original surface forms did not match (Welbl et al. (2018) used WIKIPEDIA links for masking). Indeed, in the masked version, an entity is always referred to via the same unique surface form (e.g., `MASK1` within and across documents. In the unmasked setting, on the other hand, mentions to an entity may differ (e.g., “US” vs “United States”) and they might not be retrieved by the coreference system we are employing, making the task harder for all models. Therefore, as we rely mostly on exact matching when constructing our graph for the masked case, we are more effective in recovering coreference links on the masked rather than unmasked version.6

4 Error analysis

In this section we provide an error analysis for our best single model predictions. First of all, we look at which type of questions our model performs well or poorly. There are more than 150 query types in the validation set but we filtered the three with the best and with the worst accuracy that have at least 50 supporting documents and at least 5 candidates. We show results in Table 3. We observe that questions regarding places (birth and death) are considered harder for Entity-GCN. We then inspect samples where our model fails while assigning highest likelihood and noticed two principal sources of failure i) a mismatch between what is written in WIKIPEDIA and what is annotated in WIKIDATA, and ii) a different degree of granularity (e.g., born in “London” vs “UK” could be considered both correct by a human but not when measuring accuracy). See Table 5 in the supplement material for some reported samples.

---

5 Since the test set is hidden from us, we cannot analyze this difference further.

6 Though other systems do not explicitly link matching mentions, they similarly benefit from masking (e.g., masks essentially single out spans that contain candidate answers).

---

<table>
<thead>
<tr>
<th>Model</th>
<th>unmasked</th>
<th>masked</th>
</tr>
</thead>
<tbody>
<tr>
<td>full (ensemble)</td>
<td>68.5</td>
<td>71.6</td>
</tr>
<tr>
<td>full (single)</td>
<td>65.1 ± 0.11</td>
<td>70.4 ± 0.12</td>
</tr>
<tr>
<td>GloVe with R-GCN</td>
<td>59.2</td>
<td>11.1</td>
</tr>
<tr>
<td>GloVe w/o R-GCN</td>
<td>51.2</td>
<td>11.6</td>
</tr>
<tr>
<td>No R-GCN</td>
<td>62.4</td>
<td>63.2</td>
</tr>
<tr>
<td>No relation types</td>
<td>62.7</td>
<td>63.9</td>
</tr>
<tr>
<td>No DOC–BASED</td>
<td>62.9</td>
<td>65.8</td>
</tr>
<tr>
<td>No MATCH</td>
<td>64.3</td>
<td>67.4</td>
</tr>
<tr>
<td>No COREF</td>
<td>64.8</td>
<td>–</td>
</tr>
<tr>
<td>No COMPLEMENT</td>
<td>64.1</td>
<td>70.3</td>
</tr>
<tr>
<td>Induced edges</td>
<td>61.5</td>
<td>56.4</td>
</tr>
</tbody>
</table>

Table 2: Ablation study on WIKIHOP validation set. The full model is our Entity-GCN with all of its components and other rows indicate models trained without a component of interest. We also report baselines using GloVe instead of ELMo with and without R-GCN. For the full model we report mean ± 1 std over 5 runs.
| Relation                          | Accuracy | P@2 | P@5 | Avg. \(|C_q|\) | Supports |
|----------------------------------|----------|-----|-----|----------------|----------|
| overall (ensemble)               | 68.5     | 81.0| 94.1| 20.4 \(\pm 16.6\)| 5129     |
| overall (single model)           | 65.3     | 79.7| 92.9| 20.4 \(\pm 16.6\)| 5129     |
| 3 best                           |          |     |     |                |          |
| member_of_political_party        | 85.5     | 95.7| 98.6| 5.4 \(\pm 2.4\) | 70       |
| record_label                     | 83.0     | 93.6| 99.3| 12.4 \(\pm 6.1\) | 283      |
| publisher                        | 81.5     | 96.3| 100.0| 9.6 \(\pm 5.1\) | 54       |
| 3 worst                          |          |     |     |                |          |
| place_of_birth                   | 51.0     | 67.2| 86.8| 27.2 \(\pm 14.5\) | 309      |
| place_of_death                   | 50.0     | 67.3| 89.1| 25.1 \(\pm 14.3\) | 159      |
| inception                        | 29.9     | 53.2| 83.1| 21.9 \(\pm 11.0\) | 77       |

Table 3: Accuracy and precision at K (P@K in the table) analysis overall and per query type. Avg. \(|C_q|\) indicates the average number of candidates with one standard deviation.

Secondly, we study how the model performance degrades when the input graph is large. In particular, we observe a negative Pearson’s correlation (-0.687) between accuracy and the number of candidate answers. However, the performance does not decrease steeply. The distribution of the number of candidates in the dataset peaks at 5 and has an average of approximately 20. Therefore, the model does not see many samples where there are a large number of candidate entities during training. Differently, we notice that as the number of nodes in the graph increases, the model performance drops but more gently (negative but closer to zero Pearson’s correlation). This is important as document sets can be large in practical applications. See Figure 3 in the supplemental material for plots.

5 Related work

In previous work, BiDAF (Seo et al., 2016), FastQA (Weissenborn et al., 2017), Coref-GRU (Dhingra et al., 2018), MHPGM (Bauer et al., 2018), and Weaver / Jenga (Raison et al., 2018) have been applied to multi-document question answering. The first two mainly focus on single document QA and Welbl et al. (2018) adapted both of them to work with WikIHOP. They process each instance of the dataset by concatenating all \(d \in S_q\) in a random order adding document separator tokens. They trained using the first answer mention in the concatenated document and evaluating exact match at test time. Coref-GRU, similarly to us, encodes relations between entity mentions in the document. Instead of using graph neural network layers, as we do, they augment RNNs with jump links corresponding to pairs of corefered mentions. MHPGM uses a multi-attention mechanism in combination with external commonsense relations to perform multiple hops of reasoning. Weaver is a deep co-encoding model that uses several alternating bi-LSTMs to process the concatenated documents and the query.

Graph neural networks have been shown successful on a number of NLP tasks (Marcheggiani and Titov, 2017; Bastings et al., 2017; Zhang et al., 2018a), including those involving document level modeling (Peng et al., 2017). They have also been applied in the context of asking questions about knowledge contained in a knowledge base (Zhang et al., 2018b). In Schlichtkrull et al. (2018), GCNs are used to capture reasoning chains in a knowledge base. Our work and unpublished concurrent work by Song et al. (2018) are the first to study graph neural networks in the context of multi-document QA. Besides differences in the architecture, Song et al. (2018) propose to train a combination of a graph recurrent network and an RNN encoder. We do not train any RNN document encoders in this work.

6 Conclusion

We designed a graph neural network that operates over a compact graph representation of a set of documents where nodes are mentions to entities and edges signal relations such as within and cross-document coreference. The model learns to answer questions by gathering evidence from different documents via a differentiable message passing algorithm that updates node representations based on their neighbourhood. Our model outperforms published results where ablations show substantial evidence in favour of multi-step reasoning. Moreover, we make the model fast by using pre-trained (contextual) embeddings.
References


Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In Proceedings of


Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018a. Graph convolution over pruned dependency trees improves relation extraction. In Proceedings of
A Implementation and experiments details

A.1 Architecture

See table 4 for an outline of Entity-GCN architectural detail. Here the computational steps

1. ELMo embeddings are a concatenation of three 1024-dimensional vectors resulting in 3072-dimensional input vectors \( \{ x_i \}_{i=1}^N \).

2. For the query representation \( q \), we apply 2 bi-LSTM layers of 256 and 128 hidden units to its ELMo vectors. The concatenation of the forward and backward states results in a 256-dimensional question representation.

3. ELMo embeddings of candidates are projected to 256-dimensional vectors, concatenated to the \( q \), and further transformed with a two layers MLP of 1024 and 512 hidden units in 512-dimensional query aware entity representations \( \{ \hat{x}_i \}_{i=1}^N \in \mathbb{R}^{512} \).

4. All transformations \( f \) in R-GCN-layers are affine and they do maintain the input and output dimensionality of node representations the same (512-dimensional).

5. Eventually, a 2-layers MLP with [256, 128] hidden units takes the concatenation between \( \{ h^{(L)}_i \}_{i=1}^N \) and \( q \) to predict the probability that a candidate node \( v_i \) may be the answer to the query \( q \) (see Equation 1).

During preliminary trials, we experimented with different numbers of R-GCN-layers (in the range 1-7). We observed that with WikiHop, for \( L \geq 3 \) models reach essentially the same performance, but more layers increase the time required to train them. Besides, we observed that the gating mechanism learns to keep more and more information from the past at each layer making unnecessary to have more layers than required.

<table>
<thead>
<tr>
<th>Input - q. ( { v_i }_{i=1}^N )</th>
<th>candidates ELMo 3072-dim</th>
</tr>
</thead>
<tbody>
<tr>
<td>query ELMo 3072-dim</td>
<td></td>
</tr>
<tr>
<td>2 layers bi-LSTM [256, 128]-dim</td>
<td>1 layer FF 256-dim</td>
</tr>
<tr>
<td>concatenation 512-dim</td>
<td></td>
</tr>
<tr>
<td>2 layer FF [1024, 512]-dim: ( { \hat{x}<em>i }</em>{i=1}^N )</td>
<td></td>
</tr>
<tr>
<td>3 layers R-GCN 512-dim each (shared parameters)</td>
<td></td>
</tr>
<tr>
<td>concatenation with ( q ) 768-dim</td>
<td></td>
</tr>
<tr>
<td>3 layers FF [256,128,1]-dim</td>
<td></td>
</tr>
</tbody>
</table>

Output - probabilities over \( C_q \)

Table 4: Model architecture.

A.2 Training details

We train our models with a batch size of 32 for at most 20 epochs using the Adam optimizer (Kingma and Ba, 2015) with \( \beta_1 = 0.9, \beta_2 = 0.999 \) and a learning rate of \( 10^{-4} \). To help against overfitting, we employ dropout (drop rate \( \in [0, 0.1, 0.15, 0.2, 0.25] \) (Srivastava et al., 2014) and early-stopping on validation accuracy. We report the best results of each experiment based on accuracy on validation set.
## Error analysis

In Table 5, we report three samples from WIKIHOP development set where our model might fail. In particular, we show two instances where our model presents high confidence on the answer, and one where is not. We commented these samples explaining why our model might fail in these cases.

<table>
<thead>
<tr>
<th>ID</th>
<th>WH_dev_2257</th>
<th>Gold answer</th>
<th>2003 ($p = 14.1$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query</td>
<td>inception (of) Derrty Entertainment</td>
<td><strong>Predicted answer</strong></td>
<td>2000 ($p = 15.8$)</td>
</tr>
<tr>
<td>Support 1</td>
<td>Derrty Entertainment is a record label founded by [...]. The first album released under Derrty Entertainment was Nelly’s <strong>Country Grammar</strong>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support 2</td>
<td><strong>Country Grammar</strong> is the debut single by American rapper Nelly. The song was produced by Jason Epperson. It was released in <strong>2000</strong>. [...]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(a) In this example, the model predicts the answer correctly. However, there is a mismatch between what is written in WIKIPEDIA and what is annotated in WIKIDATA. In WIKIHOP, answers are generated with WIKIDATA.

<table>
<thead>
<tr>
<th>ID</th>
<th>WH_dev_2401</th>
<th>Gold answer</th>
<th>Adolph Zukor ($p = 7.1e-4$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query</td>
<td>producer (of) Forbidden Paradise</td>
<td><strong>Predicted answer</strong></td>
<td>Jesse L. Lask ($p = 99.9$%))</td>
</tr>
<tr>
<td>Support 1</td>
<td>Forbidden Paradise is a [...] drama film produced by <strong>Famous Players-Lasky</strong> [...]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support 2</td>
<td><strong>Famous Players-Lasky</strong> Corporation was [...] from the merger of Adolph Zukor’s Famous Players Film Company [...] and the <strong>Jesse L. Lasky</strong> Feature Play Company.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b) In this sample, there is ambiguity between two entities since both are correct answers reading the passages but only one is marked as correct. The model fails assigning very high probability to only one of them.

<table>
<thead>
<tr>
<th>ID</th>
<th>WH_dev_3030</th>
<th>Gold answer</th>
<th>Scania ($p = 0.029$%))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query</td>
<td>place_of_birth (of) Erik Penser</td>
<td><strong>Predicted answer</strong></td>
<td>Eslöv ($p = 97.3$%))</td>
</tr>
<tr>
<td>Support 1</td>
<td>Nils Wilhelm Erik Penser (born August 22, 1942, in Eslöv, <strong>Skåne</strong>) is a Swedish [...]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support 2</td>
<td><strong>Skåne</strong> County, sometimes referred to as “Scania County” in English, is the [...]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(c) In this sample, there is ambiguity between two entities since the city Eslöv is located in the Scania County (English name of Skåne County). The model assigning high probability to the city and it cannot select the county.

Table 5: Samples from WIKIHOP set where Entity-GCN fails. $p$ indicates the predicted likelihood.
C Ablation study

In Figure 3, we show how the model performance goes when the input graph is large. In particular, how Entity-GCN performs as the number of candidate answers or the number of nodes increases.

Figure 3: Accuracy (blue) of our best single model with respect to the candidate set size (on the top) and nodes set size (on the bottom) on the validation set. Re-scaled data distributions (orange) per number of candidate (top) and nodes (bottom). Dashed lines indicate average accuracy.

(a) Candidates set size (x-axis) and accuracy (y-axis). Pearson’s correlation of $-0.687$ ($p < 10^{-7}$).

(b) Nodes set size (x-axis) and accuracy (y-axis). Pearson’s correlation of $-0.385$ ($p < 10^{-7}$).