Evidence Integration for Multi-Hop Reading Comprehension With Graph Neural Networks

Linfeng Song, Zhiguo Wang, Mo Yu, Yue Zhang, Radu Florian, and Daniel Gildea

Abstract—Multi-hop reading comprehension focuses on one type of factoid question, where a system needs to properly integrate multiple pieces of evidence to correctly answer a question. Previous work approximates global evidence with local coreference information, encoding coreference chains with DAG-styled GRU layers within a gated-attention reader. However, coreference is limited in providing information for rich inference. We introduce a new method for better connecting global evidence, which forms more complex graphs compared to DAGs. To perform evidence integration on our graphs, we investigate two recent graph neural networks, namely graph convolutional network (GCN) and graph recurrent network (GRN). Experiments on two standard datasets show that richer global information leads to better answers. Our approach shows highly competitive performances on these datasets without deep language models (such as ELMo).

Index Terms—Natural language processing, question answering, multi-hop reasoning, graph neural network

1 Introduction

Recent years have witnessed a growing interest in the task of machine reading comprehension. Most existing work [1], [2], [3], [4], [5], [6], [7], [8] focuses on a factoid scenario where the questions can be answered by simply considering very local information, such as one or two sentences. For example, to correctly answer a question "What causes precipitation to fall?", a QA system only needs to refer to one sentence in a passage: "... In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. ...", and the final answer "gravity" is indicated by the key words of "precipitation" and "falls".

A more challenging yet practical extension is multi-hop reading comprehension (MHRC) [9], where a system needs to properly integrate multiple pieces of evidence to correctly answer a question. Fig. 1 shows an example, which contains three associated passages, a question and several candidate choices. In order to correctly answer the question, a system has to integrate the facts "The Hanging Gardens are in Mumbai" and "Mumbai is a city in India". There are also some irrelevant facts, such as "The Hanging Gardens provide sunset views over the Arabian Sea" and "The Arabian Sea is bounded by Pakistan and Iran", which make the task more challenging, as an MHRC model has to distinguish the relevant facts from the irrelevant ones.

As a practical task, so far MHRC has received increasing research attention. One notable method, Coref-GRU [10], uses coreference information to gather richer context for each candidate. However, one main disadvantage of Coref-GRU is that the coreferences it considers are usually local to a sentence, neglecting other useful global information. In addition, the resulting DAGs are usually very sparse, thus few new facts can be inferred. The top part of Fig. 2 shows a directed acyclic graph (DAG) with only coreference edges. In particular, the two coreference edges indicate the two facts: "The Hanging Gardens provide views over the Arabian Sea" and "Mumbai is a city in India", from which we cannot indicate the ultimate fact, "The Hanging Gardens are in India", for correctly answering this instance.

We propose a general graph scheme for evidence integration, which allows information exchange beyond co-reference nodes, by allowing arbitrary degrees of the connectivity of the reference graphs. In general, we want the resulting graphs to be more densely connected so that more useful facts can be inferred. For example each edge can connect two related entity mentions, while unrelated mentions, such as "the Arabian Sea" and "India", may not be connected. In this paper, we consider three types of relations as shown in the bottom part of Fig. 2.

The first type of edges connect the mentions of the *same* entity appearing across passages or further apart in the same passage. Shown in Fig. 2, one instance connects the two "Mumbai" across the two passages. Intuitively, *same*-typed edges help to integrate global evidence related to the same entity, which are not covered by pronouns. The second type of edges connect two mentions of different entities within a context *window*. They help to pass useful evidence further across entities. For example, in the bottom graph of Fig. 2, both *window*-typed edges of ① and ⑥ help to pass evidence from "The Hanging Gardens" to "India", the answer of this

- *Linfeng Song is with the Tencent AI Lab, Bellevue, WA 98004 USA. E-mail: freesunshine0316@gmail.com.*
- *Zhiguo Wang is with the Amazon Web Service, New York, NY 10001 USA. E-mail: zgw.tomorrow@gmail.com.*
- *Mo Yu and Radu Florian are with the IBM Research, Yorktown Heights, NY 10598 USA. E-mail: {yum, raduf}@us.ibm.com.*
- *Yue Zhang is with the Westlake University, Hangzhou, Zhejiang Province 310012, China. E-mail: yue.zhang@wias.org.cn.*
- *Daniel Gildea is with the Department of Computer Science, University of Rochester, Rochester, NY 14620 USA. E-mail: gildea@cs.rochester.edu.*

Manuscript received 2 May 2019; revised 11 Dec. 2019; accepted 12 Mar. 2020. Date of publication 31 Mar. 2020; date of current version 11 Jan. 2022. (Corresponding author: Yue Zhang.)
Recommended for acceptance by A. Poulovassilis.
Digital Object Identifier no. 10.1109/TKDE.2020.2982894

[**The Hanging Gardens**], in [**Mumbai**], also known as Pherozeshah Mehta Gardens, are terraced gardens ... [**They**] provide sunset views over the [**Arabian Sea**] ...

[**Mumbai**] (also known as Bombay, the official name until 1995) is the capital city of the Indian state of Maharashtra. [**It**] is the most populous city in [**India**] ...

The [**Arabian Sea**] is a region of the northern Indian Ocean bounded on the north by [**Pakistan**] and [**Iran**], on the west by northeastern [**Somalia**] and the Arabian Peninsula, and on the east by ...

Q: (The Hanging gardens, country, ?)
Options: Iran, India, Pakistan, Somalia, ...

Fig. 1. An example from WikiHop [9], where some relevant entity mentions and their anaphoric pronouns are highlighted.

instance. Besides, *window*-typed edges enhance the relations between local mentions that can be missed by the sequential encoding baseline. Finally, *coreference*-typed edges, each of which connects a pronoun and the corresponding mention, are further complimentary to the previous two types, and thus we also include them.

Since our generated graphs are complex and can have cycles, making it difficult to directly apply a DAG network (e.g. the structure of Coref-GRU), we adopt graph neural networks [11], which can encode arbitrary graphs. In particular, we choose graph convolutional network (GCN) and graph recurrent network (GRN), as they have been shown successful on encoding semantic graphs [12], dependency graphs [13], [14], [15], [16], raw texts [17] and other complex data structures [18], [19], [20].

Given an instance containing several passages and a list of candidates, we first use NER and coreference resolution tools to obtain entity mentions, and then create a graph out of the mentions and relevant pronouns. As the next step, evidence integration is executed on the graph by adopting a graph neural network on top of a sequential layer. The sequential layer learns local representation for each mention, while the graph network learns a global representation. The answer is decided by matching the representations of the mentions against the question representation.

Experiments on WikiHop [9] show that our created graphs are highly useful for MHRC. On the hold-out testset, it achieves a highly-competitive accuracy of 65.4 percent. In addition, our experiments show that the questions and answers are dramatically better connected on our graphs than on the coreference DAGs, if we map the questions on graphs using the question subject. Our experiments also show a positive relation between graph connectivity and end-to-end accuracy.

On the testset of ComplexWebQuestions [21], our method also achieves better results than all published numbers. To our knowledge, we are among the first to investigate graph neural networks on reading comprehension tasks. Our code is available at https://github.com/freesunshine0316/MHQA.

2 Related Work

Question Answering With Multi-Hop Reasoning. Multi-hop reasoning is an important ability for dealing with difficult cases in question answering [22], [23]. Most existing work on multi-hop QA focuses on hopping over knowledge bases or tables [24], [25], [26], thus the problem is reduced to deduction on a readily-defined structure with known relations. In contrast, we study multi-hop QA on textual data and we introduce an effective approach for creating evidence integration graph structures over the textual input for solving our problems. Previous work [7], [27] studying multi-hop QA on text does not create reference structures. In addition, they only evaluate their models on a simple task [28] with a very limited vocabulary and passage length. Our work is fundamentally different from theirs by modeling structures over the input, and we evaluate our models on more challenging tasks.

Recent work starts to exploit ways for creating structures from inputs. Talmor and Berant [21] build a two-level computation tree over each question, where the first-level nodes are sub-questions and the second-level node is a composition operation. The answers for the sub-questions are first generated, and then combined with the composition operation. They predefine two composition operations, which makes it not general enough for other QA problems. Dhingra *et al.* [10] create DAGs over passages with coreference. The DAGs are then encoded using a DAG recurrent network. Our work follows the second direction by creating reasoning graphs on the passage side. However, we consider more types of relations than coreference, making a thorough study on evidence integration. Besides, we also investigate recent graph neural networks (namely GCN and GRN) on this problem.

Being released in the same period and similar to our work, Cao *et al.* [29] also proposed a graph-based model using a GCN for multi-hop reasoning. In addition to the graph neural network being used, there are also differences on graph construction: (1) they only consider the question subject and the candidates as graph nodes, while we consider all named entities (including both the question subject and the candidates), and (2) they further distinguish inter- and intra-paragraph edges with the same relation, while we consider them as the same type. There are also later efforts that follow this path by explicitly creating graphs for reasoning. Tu *et al.* [30] further propose a heterogeneous graph network to separately model different types of graph nodes. Cao *et al.* [31] enrich a graph-based model with rich features of POS and NER tags, and they also add a bi-directional attention module to their model.

Question Answering Over Multiple Passages. Recent efforts in open-domain QA start to generate answers from multiple passages instead of a single passage. However, most existing work on multi-passage QA selects the most relevant passage for answering the given question, thus reducing the problem to single-passage reading comprehension [32], [33], [34], [35], [36]. Our method is fundamentally different by truly leveraging multiple passages.

A few multi-passage QA approaches merge evidence from multiple passages before selecting an answer [37], [38], [39]. Similar to our work, they combine evidences from multiple passages, thus fully utilizing input passages. The key difference is that their approaches focus on how the contexts of a single answer candidate from different passages could cover different aspects of a complex question, while our approach studies how to properly integrate the related evidence of an answer candidate, some of which come from the contexts of different entity mentions. This increases the difficulty, since those contexts do not co-occur with the candidate answer nor the question. When a piece of evidence does not co-occur with the answer candidate, it is usually difficult for these methods

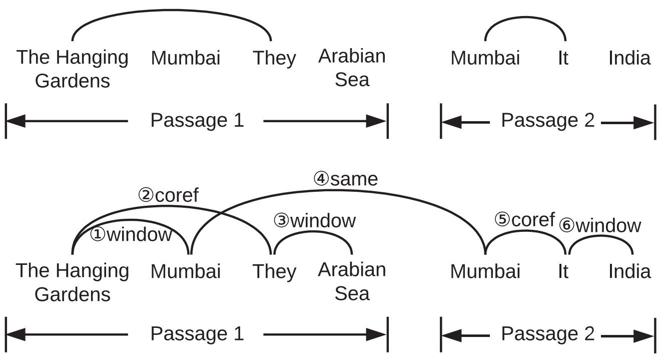

Fig. 2. A DAG generated by Dhingra *et al.* [10] (top) and a graph by considering all three types of edges (bottom) on the example in Fig. 1.

to integrate the evidence. This is also demonstrated by our empirical comparison, where our approach shows much better performance than combining only the evidence of the same entity mentions.

3 Baseline

As shown in Fig. 3, we introduce two baselines, which are inspired by Dhingra *et al.* [10]. The first baseline, *Local*, uses a standard BiLSTM layer (shown in the green dotted box), where inputs are first encoded with a BiLSTM layer, and then the representation vectors for the mentions in the passages are extracted, before being matched against the question for selecting an answer. The second baseline, *Coref LSTM*, differs from *Local* by replacing the BiLSTM layer with a DAG LSTM layer (shown in the orange dotted box) for encoding additional coreference information, as proposed by Dhingra *et al.* [10].

3.1 *Local*: BiLSTM Encoding

Given a list of relevant passages, we first concatenate them into one large passage $p_1, p_2 \dots p_N$, where each p_i is a passage word and x_{p_i} is the embedding of it. It adopts a BiLSTM to encode the passage:

$$\overleftarrow{h}^i_p = \mathrm{LSTM}(\overleftarrow{h}^{i+1}_p, x_{p_i})$$
$$\overrightarrow{h}^i_p = \mathrm{LSTM}(\overrightarrow{h}^{i-1}_p, x_{p_i}).$$

Each hidden state contains the information of its local context. Similarly, the question words $q_1, q_2 \dots q_M$ are first converted into embeddings $x_{q_1}, x_{q_2} \dots x_{q_M}$ before being encoded by another BiLSTM:

$$\overleftarrow{h}^j_q = \mathrm{LSTM}(\overleftarrow{h}^{j+1}_q, x_{q_j})$$
$$\overrightarrow{h}^j_q = \mathrm{LSTM}(\overrightarrow{h}^{j-1}_q, x_{q_j}).$$

3.2 *Coref LSTM*: DAG LSTM Encoding With Conference Knowledge

Taking the passage word embeddings $x_{p_1}, \dots x_{p_N}$ and coreference information as the input, the DAG LSTM layer encodes each input word embedding (such as x_{p_j}) with the following gated operations:[1]

1. Only the forward direction is shown for space consideration

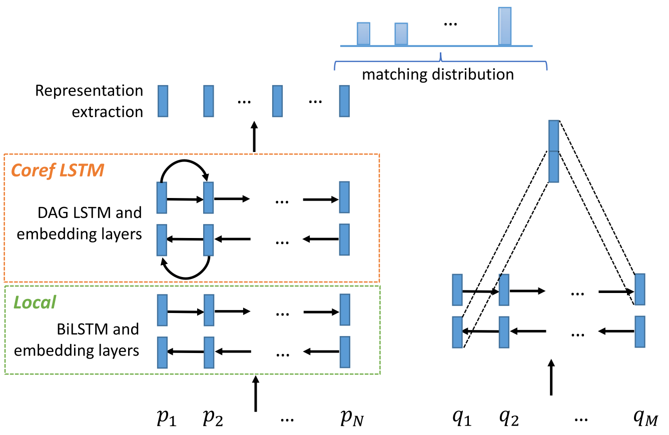
matching distribution
Representation extraction
Coref LSTM
DAG LSTM and embedding layers
Local
BiLSTM and embedding layers
p_1
p_2
p_N
q_1
q_2
q_M

Fig. 3. Baselines. The upper dotted box is a DAG LSTM layer with addition coreference links, while the bottom one is a typical BiLSTM layer. Either layer is used.

$$i_j = \sigma\left(W_i x_{p_j} + U_i \sum_{i \in \mathbb{N}(j)} \overrightarrow{h}^i_p + b_i\right)$$
$$o_j = \sigma\left(W_o x_{p_j} + U_o \sum_{i \in \mathbb{N}(j)} \overrightarrow{h}^i_p + b_o\right)$$
$$f_{i,j} = \sigma(W_f x_{p_j} + U_f \overrightarrow{h}^i_p + b_f)$$
$$u_j = \sigma\left(W_u x_{p_j} + U_u \sum_{i \in \mathbb{N}(j)} \overrightarrow{h}^i_p + b_u\right)$$
$$\overrightarrow{c}^j_p = i_j \odot u_j + \sum_{i \in \mathbb{N}(j)} f_{i,j} \odot \overrightarrow{c}^i_p$$
$$\overrightarrow{h}^j_p = o_j \odot \tanh(\overrightarrow{c}^j_p),$$

$\mathbb{N}_j$ represents all preceding words of p_j in the DAG, i_j, o_j and $f_{i,j}$ are the input, output and forget gates, respectively. W_x, U_x and b_x ($x \in \{i, o, f, u\}$) are model parameters.

3.3 Representation Extraction

After encoding both the passage and the question, we obtain a representation vector for each entity mention ϵ_k, spanning from k_i to k_j, by concatenating the hidden states of its start and end positions, before they are correlated with a fully connected layer:

$$h^k_\epsilon = W_1[\overleftarrow{h}^{k_i}_p; \overrightarrow{h}^{k_i}_p; \overleftarrow{h}^{k_j}_p; \overrightarrow{h}^{k_j}_p] + b_1, \tag{1}$$

where W_1 and b_1 are model parameters for compressing the concatenated vector. Note that the current multi-hop reading comprehension datasets all focus on the situation where the answer is a named entity. Similarly, the representation vector for the question is generated by concatenating the hidden states of its first and last positions:

$$h_q = W_2\left[\overleftarrow{h}^1_q; \overrightarrow{h}^1_q; \overleftarrow{h}^M_q; \overrightarrow{h}^M_q\right] + b_2, \tag{2}$$

where W_2 and b_2 are also model parameters.

3.4 Attention-Based Matching

Given the representation vectors for the question and the entity mentions in the passages, an additive attention model

[40] is adopted by treating all entity mention representations and the question representation as the memory and the query, respectively. In particular, the probability for a candidate c_φ being the answer given input X is calculated by summing up all the occurrences of c_φ across the input passages:

$$p(c_\varphi|X) = \frac{\sum_{k\in\mathcal{N}_{c_\varphi}} \alpha_k}{\sum_{k'\in\mathcal{N}_c} \alpha_{k'}}, \tag{3}$$

where $\mathcal{N}_{c_\varphi}$ and $\mathcal{N}_c$ represent all occurrences of the candidate c_φ and all occurrences of all candidates, respectively. Previous work [35] shows that summing the probabilities over all occurrences of the same entity mention is important for the multi-passage scenario. α_k is the attention score for the entity mention ϵ_k, calculated by an additive attention model shown below:

$$e_0^k = \boldsymbol{v}_\alpha^T \tanh(\boldsymbol{W}_\alpha \boldsymbol{h}_\epsilon^k + \boldsymbol{U}_\alpha \boldsymbol{h}_q + \boldsymbol{b}_\alpha) \tag{4}$$

$$\alpha_k = \frac{\exp(e_0^k)}{\sum_{k'\in\mathcal{N}} \exp(e_0^{k'})}, \tag{5}$$

where $\boldsymbol{v}_\alpha$, $\boldsymbol{W}_\alpha$, $\boldsymbol{U}_\alpha$ and $\boldsymbol{b}_\alpha$ are model parameters, and $\mathcal{N}$ represents all occurrences of all entities.

3.5 Comparison With Dhingra *et al.* [10]

The Coref-GRU model [10] is based on the gated-attention reader (GA reader) [6], which is designed for the cloze-style reading comprehension task [1], where *one* token is selected from the input passages as the answer for each instance. To adapt their model for the WikiHop benchmark, where an answer candidate can contain multiple tokens, they first generate a probability distribution over the passage tokens with GA reader, and then compute the probability for each candidate c by aggregating the probabilities of all passage tokens that appear in c and renormalizing over the candidates.

In addition to using LSTM instead of GRU, the main difference between our two baselines and Dhingra *et al.* [10] is that our baselines consider each candidate as a whole unit no matter whether it contains multiple tokens or not. This makes our models more effective on the datasets containing phrasal answer candidates.

4 Evidence Integration With Graph Neural Networks

Over the representation vectors for a question and the corresponding entity mentions, we build an evidence integration graph of the entity mentions by connecting relevant mentions with edges, and then integrating relevant information for each graph node (entity mention) with a graph recurrent network (GRN) [12], [17] or a graph convolutional network (GCN) [41].[2] Fig. 4 shows the overall procedure of our approach.

4.1 Graph Construction

As a first step, we create a graph from a list of input passages. The entity mentions within the passages are taken as the graph nodes. They are automatically generated by NER and coreference annotators, so that each graph node is either an entity mention or a pronoun representing an entity. We then create a graph by ensuring that edges between two nodes follow the situations below:

- They are occurrences of the *same* entity mention across passages or with a distance larger than a threshold τ_L when being in the same passage.
- One is an entity mention and the other is its *coreference* pronoun. This information is automatically generated by a coreference resolution toolkit.
- Between two mentions of different entities in the same passage within a *window* threshold of τ_S.

Between every two entities that satisfy the situations above, we make two edges in opposite directions. As a result, each generated graph can also be considered as an undirected graph.

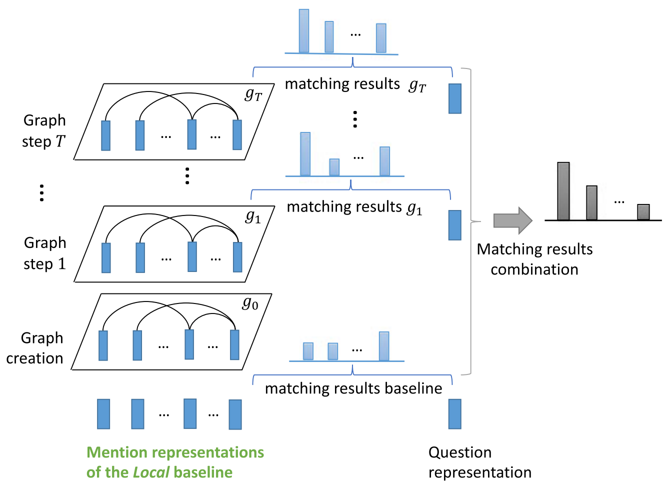

Fig. 4. Model framework.

4.2 Evidence Integration With Graph Encoding

Tackling multi-hop reading comprehension requires inferring on global context. As the next step, we merge related information through the three types of edges just created. We investigate two recent graph networks: GRN and GCN.

Graph Recurrent Network (GRN). GRN models a graph as a single state, performing recurrent information exchange between graph nodes through graph state transitions. Formally, given a graph $G = (V, E)$, a hidden state vector s^k is created to represent each entity mention $\epsilon_k \in V$. The state of the graph can thus be represented as:

$$\boldsymbol{g} = \{\boldsymbol{s}^k\}|\epsilon_k \in V,$$

In order to integrate non-local evidence among nodes, information exchange between neighborhooding nodes is performed through recurrent state transitions, leading to a sequence of graph states $\boldsymbol{g}_0, \boldsymbol{g}_1, \ldots, \boldsymbol{g}_T$, where $\boldsymbol{g}_T = \{\boldsymbol{s}_T^k\}|\epsilon_k \in V$ and T is a hyperparameter representing the number of graph state transition decided by a development experiment. For initial state $\boldsymbol{g}_0 = \{\boldsymbol{s}_0^k\}|\epsilon_k \in V$, we initialize each $\boldsymbol{s}_0^k$ by:

$$\boldsymbol{s}_0^k = \boldsymbol{W}_3[\boldsymbol{h}_\epsilon^k; \boldsymbol{h}_q] + \boldsymbol{b}_3, \tag{6}$$

where $\boldsymbol{h}_\epsilon^k$ is the corresponding representation vector of entity mention ϵ_k, calculated by Equation 1. $\boldsymbol{h}_q$ is the question representation. $\boldsymbol{W}_3$ and $\boldsymbol{b}_3$ are model parameters.

2. We consider investigating other graph neural networks (such as Beck *et al.* [42]) as a future work.

A gated recurrent neural network is used to model the state transition process. In particular, the transition from g_{t-1} to g_t consists of a hidden state transition for each node. At each step t, direct information exchange is conducted between a node and all its neighbors via the following LSTM [43] operations:

$$\begin{aligned}
\boldsymbol{i}_t^k &= \sigma(\boldsymbol{W}_i \boldsymbol{m}_t^k + \boldsymbol{b}_i) \\
\boldsymbol{o}_t^k &= \sigma(\boldsymbol{W}_o \boldsymbol{m}_t^k + \boldsymbol{b}_o) \\
\boldsymbol{f}_t^k &= \sigma(\boldsymbol{W}_f \boldsymbol{m}_t^k + \boldsymbol{b}_f) \\
\boldsymbol{u}_t^k &= \sigma(\boldsymbol{W}_u \boldsymbol{m}_t^k + \boldsymbol{b}_u) \\
\boldsymbol{c}_t^k &= \boldsymbol{f}_t^k \odot \boldsymbol{c}_{t-1}^k + \boldsymbol{i}_t^k \odot \boldsymbol{u}_t^k \\
\boldsymbol{s}_t^k &= \boldsymbol{o}_t^k \odot \tanh(\boldsymbol{c}_t^k),
\end{aligned} \tag{7}$$

where $\boldsymbol{c}_t^k$ is the cell vector to record memory for $\boldsymbol{s}_t^k$, and $\boldsymbol{i}_t^k$, $\boldsymbol{o}_t^k$ and $\boldsymbol{f}_t^k$ are the input, output and forget gates, respectively. $\boldsymbol{W}_x$ and $\boldsymbol{b}_x$ ($x \in \{i, o, f, u\}$) are model parameters. $\boldsymbol{m}_t^k$ is the sum of the neighborhood hidden states for the node ϵ_k:[3]

$$\boldsymbol{m}_t^k = \sum_{i \in \mathbb{N}(k)} \boldsymbol{s}_{t-1}^i, \tag{8}$$

$\mathbb{N}(k)$ represents the set of all neighbors of ϵ_k.

Graph Convolutional Network (GCN). GCN is a convolution-based alternative to GRN for encoding graphs. Similar with GRN, encoding with a GCN model consists of two main steps: state initialization and state update. For state initialization, GCN adopts the same approach as with GRN by initializing from the representations vectors of entity mentions, as shown in Equation (6). The main difference between GCN and GRN is the way for updating node states. GRN adopts gated operations (shown in Equation (7)), while GCN uses linear transportation with ReLU as the activation function:

$$\boldsymbol{s}_t^k = \mathrm{ReLU}(\boldsymbol{W}_g \boldsymbol{m}_t^k + \boldsymbol{b}_g), \tag{9}$$

where $\boldsymbol{m}_t^k$ is also the sum of the neighborhood hidden states defined in Equation (8). $\boldsymbol{W}_g$ and $\boldsymbol{b}_g$ are model parameters.

4.3 Matching and Combination

After evidence integration, we match the hidden states at each graph encoding step with the question representation using the same additive attention mechanism introduced in the Baseline section. In particular, for each entity ϵ_k, the matching results for the baseline and each graph encoding step t are first generated, before being combined using a weighted sum to obtain the overall matching result:

$$e_t^k = \boldsymbol{v}_{a_t}^T \tanh(\boldsymbol{s}_t^k \boldsymbol{W}_{a_t} + \boldsymbol{h}_q \boldsymbol{U}_{a_t} + \boldsymbol{b}_{a_t}) \tag{10}$$

$$e^k = \boldsymbol{w}_c \odot [e_0^k, e_1^k, \ldots, e_T^k] + b_c, \tag{11}$$

where e_0^k is the baseline matching result for ϵ_k, e_t^k is the matching results after t steps, and T is the total number of graph encoding steps. $\boldsymbol{W}_{a_t}$, $\boldsymbol{U}_{a_t}$, $\boldsymbol{v}_{a_t}$, $\boldsymbol{b}_{a_t}$, $\boldsymbol{w}_c$ and b_c are model parameters. In addition, a probability distribution is calculated from the overall matching results using softmax, similar to Equation (5). Finally, probabilities that belong to the same entity mention are merged to obtain the final distribution, as shown in Equation (3).

3. We tried distinguishing neighbors by different types of edges, but it does not improve the performance.

5 Training

We train both the baseline and our models using the cross-entropy loss:

$$l = -\log p(c_\varphi^* | X; \theta), \tag{12}$$

where c_φ^* is ground-truth answer, X and θ are the input and model parameters, respectively. Adam [44] with a learning rate of 0.001 is used as the optimizer. Dropout with rate 0.1 and a $l2$ normalization weight of 10^{-8} are used during training.

6 Experiments on WikiHop

In this section, we study the effectiveness of rich types of edges and the graph encoders using the WikiHop [9] dataset.

6.1 Data

The dataset contains around 51K instances, including 44 K for training, 5K for development and 2.5K for held-out testing. Each instance consists of a question, a list of associated passages, a list of candidate answers and a correct answer. One example is shown in Fig. 1. We use Stanford CoreNLP [45] to obtain coreference and NER annotations. Then the entity mentions, pronoun coreferences and the provided candidates are taken as graph nodes to create an evidence graph. The distance thresholds (τ_L and τ_S, in Section 4.1) for making *same* and *window* typed edges are set to 200 and 20, respectively.

6.2 Settings

We study the model behavior on the WikiHop devset, choosing the best hyperparameters for online system evaluation on the final holdout testset. Our word embeddings are initialized from the 300-dimensional pretrained Glove word embeddings [46] on Common Crawl, and are not updated during training.

For model hyperparameters, we set the graph state transition number as 3 according to development experiments. Each node takes information from at most 200 neighbors, where *same* and *coref* typed neighbors are kept first. The hidden vector sizes for both bidirectional LSTM and GRN layers are set to 300.

6.3 Development Experiments

Fig. 5 shows the devset performances of our model using GRN or GCN with different transition steps. It shows the baseline performances when transition step is 0. The performances go up for both models when increasing the transition step to 3. Further increasing the transition step leads to a slight decrease in performance. One reason can be that executing more transition steps may also introduce more noise through richly connected edges. We set the transition step to 3 for all remaining experiments. GRN shows slightly better performances than GCN with the increase of transition steps. However, the gap is not very significant, with the main reason being the small number of steps (up to 4).

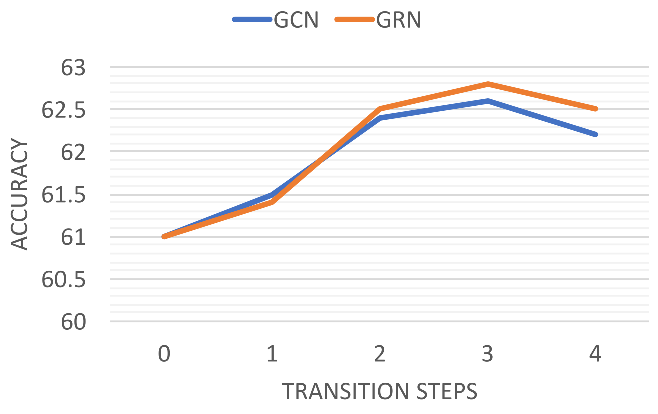

Fig. 5. Dev performances of different transition steps.

TABLE 1
Main Results on WikiHop, Where Systems With † Indicates Using a Deep Language Model or a Gigantic Model Architecture

Model	Dev	Test
GA w/ GRU [10]	54.9	–
GA w/ Coref-GRU [10]	56.0	59.3
Local	61.0	–
Local-2L	61.3	–
Coref LSTM	61.4	–
Coref GRN	61.4	–
Fully-Connect-GRN	61.3	–
MHQA-GCN	62.6	–
MHQA-GRN	**62.8**	**65.4**
Reasoning Chain [47]†	–	76.5
DynSAN [48]†	–	71.4

6.4 Main Results

Table 1 shows the main comparison results with existing work. *GA w/ GRU* and *GA w/ Coref-GRU* correspond to Dhingra *et al.* [10], and their reported numbers are copied. The former is their baseline, a gated-attention reader [6], and the latter is their proposed method.

For our baselines, *Local* and *Local-2L* encode passages with a BiLSTM and a 2-layer BiLSTM, respectively, both only capture local information for each mention. We introduce *Local-2L* for better comparison, as our models have more parameters than *Local*. *Coref LSTM* is another baseline, encoding passages with coreference annotations by a DAG LSTM (Section 3.2). This is a reimplementation of Dhingra *et al.* [10] based on our framework. *Coref GRN* is another baseline that encodes coreferences with GRN. It is for contrasting coreference DAGs with our evidence integration graphs. *MHQA-GCN* and *MHQA-GRN* correspond to our evidence integration approaches via graph encoding, adopting GCN and GRN for graph encoding, respectively.

First, even our *Local* show a much higher accuracy compared with *GA w/ GRU* and *GA w/ Coref-GRU*. This is because our models are more compatible with the evaluated dataset. In particular, *GA w/ GRU* and *GA w/ Coref-GRU* calculate the probability for each candidate by summing up the probabilities of all tokens within the candidate. As a result, they cannot handle phrasal candidates very well, especially for the overlapping candidates, such as “New York” and “New York City”. On the other hand, we consider each candidate answer as a single unit, and does not suffer from this issue. As a reimplementation of their idea, *Coref LSTM* only shows 0.4 points gains over *Local*, a stronger baseline than *GA w/ GRU*. On the other hand, *MHQA-GCN* and *MHQA-GRN* are 1.6 and 1.8 points more accurate than *Local*, respectively. Our *Local* baseline achieves a decent number without the help of explicit multi-hop operations. The reason, as mentioned recently [49], is that many easy instances can be inferred from the entity types of candidates. But, our model still largely improves the performance by tackling more difficult instances.

The comparisons below help to further pinpoint the advantage of our approach: *MHQA-GRN* is 1.4 points better than *Coref GRN* , while *Coref GRN* gives a comparable performance with *Coref LSTM*. Both comparisons show that our evidence graphs are the main reason for achieving the 1.8-points improvement, and it is mainly because our evidence graphs are better connected than coreference DAGs and are more suitable for integrating relevant evidence. *Local-2L* is not significantly better than *Local*, meaning that simply introducing more parameters does not help.

In addition to the systems above, we introduce *Fully-Connect-GRN* for demonstrating the effectiveness of our evidence graph creating approach. *Fully-Connect-GRN* creates fully connected graphs out of the entity mentions, before encoding them with GRN. Within each fully connected graph, the question is directly connected with the answer. However, fully connected graphs are brute-force connections, and are not representative for integrating related evidence. *MHQA-GRN* is 1.5 points better than *Fully-Connect-GRN*, while questions and answers are more directly connected (with distance 1 for all cases) by *Fully-Connect-GRN*. The main reason can be that our evidence graphs only connect related entity mentions, making our models easier to learn how to integrate evidence. On the other hand, there are barely learnable patterns within fully connected graphs. More analyses on the relation between graph connectivity and end-to-end performance will be shown in later paragraphs.

We observe some recent papers showing better results on the leaderboard.[4] The last group of Table 1 shows the top two published non-ensemble models, where *Reasoning Chain* uses a BERT [50] model, and *DynSAN* adopts a very deep model architecture that needs 4 12GB GPUs for training. Our main contribution is studying an evidence integration approach, which is orthogonal to the contribution of BERT and other deep model-framework design. We will investigate these in a future version.

6.5 Analysis

Effectiveness of Edge Types. Table 2 shows the ablation study of different types of edges that we introduce for evidence integration. The first group shows the situations where one type of edges are removed. In general, there is a large performance drop by removing any type of edges. The reason can be that the connectivity of the resulting graphs is reduced, thus fewer facts can be inferred. Among all these types, removing *window*-typed edges causes the least performance drop. One possible reason is that some information captured by them has

4. https://qangaroo.cs.ucl.ac.uk/leaderboard.html, as of Dec. 11th, 2019.

TABLE 2
Ablation Study on Different Types of Edges Using GRN as the Graph Encoder

Edge type	Dev
all types	62.8
w/o same	61.9
w/o coref	61.7
w/o window	62.4
only same	61.6
only coref	61.4
only window	61.1

been well captured by sequential encoding. However, *window*-typed edges are still useful, as they can help passing evidence through to further nodes. Take Fig. 2 as an example, two *window*-typed edges help to pass information from "The Hanging Gardens" to "India". The other two types of edges are slightly more important than *window*-typed edges. Intuitively, they help to gather more global information than *window*-typed edges, thus learn better representations for entities by integrating contexts from their occurrences and co-references.

The second group of Table 2 shows the model performances when only one type of edges are used. None of the performances with single-typed edges are significantly better than the *Local* baseline, whereas the combination of all types of edges achieves a much better accuracy (1.8 points) than *Local*. This indicates the importance of evidence integration over better connected graphs. We show more detailed quantitative analyses later on. The numbers generally demonstrate the same patterns as the first group. In addition, *only same* is slightly better than *only coref*. It is likely because some coreference information can also be captured by sequential encoding.

Distance. Fig. 6 shows the percentage distribution of distances between a question and its closest answer when either all types of edges are adopted or only coreference edges are used. The subject of each question[5] is used to locate the question on the corresponding graph.

When all types of edges are adopted, the questions and the answers for more than 90 percent of the development instances are connected, and the question-and-answer distances for more than 70 percent are within 3. On the other hand, the instances with distances longer than 4 only count for 10 percent. This can be the reason why performances do not increase when more than 3 transition steps are performed in our model. The advantage of our approach can be shown by contrasting the distance distributions over graphs generated either by the baseline or by our approach.

We further evaluate both approaches on a subset of the development instances, where the answer-and-question distance is at most 3 in our graph. The accuracies of *Coref LSTM* and *MHQA-GRN* on this subset are 61.1 and 63.8, respectively. Comparing with the performances on the whole devset (61.4 vs 62.8), the performance gap on this subset is increased by 1.3 points. This indicates that our approach can better handle these "relatively easy" reasoning tasks.

5. As shown in Fig. 1, each question has a subject, a relation and asks for the object.

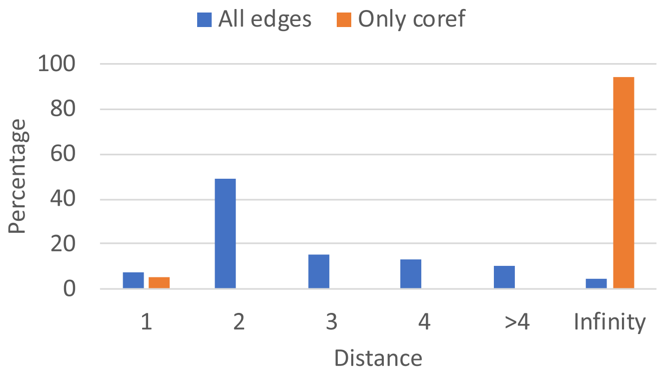

Fig. 6. Distribution of distances between a question and an answer on the DEVSET.

However, as shown in Fig. 5, instances that require large reasoning steps are still challenging to our approach.

7 EXPERIMENTS ON COMPLEXWEBQUESTIONS

In this section, we conduct experiments on the newly released ComplexWebQuestions version 1.1 [21] for better evaluating our approach. Compared with WikiHop, where the complexity is implicitly specified in the passages, the complexity of this dataset is explicitly specified on the question side. One example question is "What city is the birthplace of the author of 'Without end"'. A two-step reasoning is involved, with the first step being "the author of 'Without end"' and the second being "the birthplace of x". x is the answer of the first step.

In this dataset, web snippets (instead of passages as in WikiHop) are used for extracting answers. The baseline of Talmor and Berant [21] (*SimpQA*) only uses a full question to query the web for obtaining relevant snippets, while their model (*SplitQA*) obtains snippets for both the full question and its sub-questions. With all the snippets, *SplitQA* models the QA process based on a computation tree[6] of the full question. In particular, they first obtain the answers for the sub-questions, and then integrate those answers based on the computation tree. In contrast, our approach creates a graph from all the snippets, thus the succeeding evidence integration process can join all associated evidence.

Main Results. As shown in Table 3, similar to the observations in WikiHop, both *MHQA-GRN* and *MHQA-GCN* achieve large improvements over *Local*, and *MHQA-GRN* gives slightly better accuracy. Both the baselines and our models use all web snippets, but *MHQA-GRN* and *MHQA-GCN* further consider the structural relations among entity mentions. *SplitQA* achieves 0.5 percent improvement over *SimpQA*.[7] Our *Local* baseline is comparable with *SplitQA* and our graph-based models contribute a further 2 percent improvement over *Local*. This indicates that considering structural information on passages is important for the dataset.

Analysis. To deal with complex questions that require evidence from multiple passages to answer, previous work

6. A computation tree is a special type of semantic parse, which has two levels. The first level contains sub-questions and the second level is a composition operation.

7. Upon the submission time, the authors of ComplexWebQuestions have not reported testing results for the two methods. To make a fair comparison we compare the devset accuracy.

TABLE 3
Results on the ComplexWebQuestions Dataset

Model	Dev	Test
SimpQA	30.6	–
SplitQA	31.1	–
Local	31.2	28.1
MHQA-GCN	32.8	–
MHQA-GRN w/ only same	32.2	–
MHQA-GRN	**33.2**	**30.1**
SplitQA w/ additional labeled data	35.6	34.2

[37], [38], [39] collect evidence from occurrences of an entity in different passages. The above methods correspond to a special case of our method, i.e. MHQA with only the *same*-typed edges. From Table 3, our method gives 1 point increase over *MHQA-GRN w/ only same*, and it gives more increase in WikiHop (comparing *all types* with *only same* in Table 2). Both results indicate that our method could capture more useful information for multi-hop QA tasks, compared to the methods developed for previous multi-passage QA tasks. This is likely because our method integrates not only evidences for an entity but also these for other related entities.

The leaderboard reports *SplitQA* with additional sub-question annotations and gold answers for sub-questions. These pairs of sub-questions and answers are used as additional data for training *SplitQA*. The above approach relies on annotations of ground-truth answers for sub-questions and semantic parses, thus is not practically useful in general. However, the results have additional value since it can be viewed as an upper bound of *SplitQA*. Note that the gap between this upper bound and our *MHQA-GRN* is small, which further proves that larger improvement can be achieved by introducing structural connections on the passage side to facilitate evidence integration.

8 Conclusion

We have introduced a new approach for tackling multi-hop reading comprehension (MHRC), with a graph-based evidence integration process. Given a question and a list of passages, we first connect related evidence in reference passages into a graph, and then adopt recent graph neural networks to encode resulted graphs for performing evidence integration. Results show that the three types of edges are useful on combining global evidence and that the graph neural networks are effective on encoding complex graphs resulted by the first step. Our approach shows highly competitive performances on two standard MHRC datasets.

Acknowledgments

The authors would like to thank Salim Roukos from IBM Research for his helpful suggestions. They are also grateful for the constructive reviews from the action editor and all reviewers. The work of Yue Zhang was supported by NSFC under Grant 61976180. The work of Daniel Gildea was supported by NSF under Grant IIS-1813823.

References

[1] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, “Teaching machines to read and comprehend,” in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 1693–1701.
[2] S. Wang and J. Jiang, “Machine comprehension using match-LSTM and answer pointer,” in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1693–1701.
[3] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, “Bidirectional attention flow for machine comprehension,” in *Proc. Int. Conf. Learn. Representations*, 2017.
[4] Z. Wang, H. Mi, W. Hamza, and R. Florian, “Multi-perspective context matching for machine comprehension,” 2016, *arXiv:1612.04211*.
[5] D. Weissenborn, G. Wiese, and L. Seiffe, “Making neural QA as simple as possible but not simpler,” in *Proc. 21st Conf. Comput. Natural Lang. Learn.*, 2017, pp. 271–280.
[6] B. Dhingra, H. Liu, Z. Yang, W. Cohen, and R. Salakhutdinov, “Gated-attention readers for text comprehension,” in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics*, 2017, pp. 1832–1846.
[7] Y. Shen, P.-S. Huang, J. Gao, and W. Chen, “Reasonet: Learning to stop reading in machine comprehension,” in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2017, pp. 1047–1055.
[8] C. Xiong, V. Zhong, and R. Socher, “Dynamic coattention networks for question answering,” in *Proc. Int. Conf. Learn. Representations*, 2017
[9] J. Welbl, P. Stenetorp, and S. Riedel, “Constructing datasets for multi-hop reading comprehension across documents,” *Trans. Assoc. Comput. Linguistics*, vol. 6, pp. 287–302, 2018.
[10] B. Dhingra, Q. Jin, Z. Yang, W. Cohen, and R. Salakhutdinov, “Neural models for reasoning over multiple mentions using coreference,” in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Human Lang. Technol.*, 2018, pp. 42–48.
[11] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, “The graph neural network model,” *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009.
[12] L. Song, Y. Zhang, Z. Wang, and D. Gildea, “A graph-to-sequence model for amr-to-text generation,” in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, 2018, pp. 1616–1626.
[13] J. Bastings, I. Titov, W. Aziz, D. Marcheggiani, and K. Simaan, “Graph convolutional encoders for syntax-aware neural machine translation,” in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2017, pp. 1957–1967.
[14] D. Marcheggiani and I. Titov, “Encoding sentences with graph convolutional networks for semantic role labeling,” in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2017, pp. 1506–1515.
[15] S. Vashishth, S. S. Dasgupta, S. N. Ray, and P. Talukdar, “Dating documents using graph convolution networks,” in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, 2018, pp. 1605–1615.
[16] L. Song, Y. Zhang, Z. Wang, and D. Gildea, “N-ary relation extraction using graph-state LSTM,” in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2018, pp. 2226–2235.
[17] Y. Zhang, Q. Liu, and L. Song, “Sentence-state LSTM for text representation,” in *Proc. 56th Annual Meeting Assoc. Comput. Linguistics*, 2018, pp. 317–327.
[18] D. Zhu, Z. Zhang, P. Cui, and W. Zhu, “Robust graph convolutional networks against adversarial attacks,” in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2019, pp. 1399–1407.
[19] J. Ma, P. Cui, K. Kuang, X. Wang, and W. Zhu, “Disentangled graph convolutional networks,” in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 4212–4221.
[20] X. Wang *et al.*, “Heterogeneous graph attention network,” in *Proc. World Wide Web Conf.*, 2019, pp. 2022–2032.
[21] A. Talmor and J. Berant, “The web as a knowledge-base for answering complex questions,” in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Human Lang. Technol.*, 2018, pp. 641–651.
[22] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, “SQuAD: 100,000+ questions for machine comprehension of text,” in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2016, pp. 2383–2392.
[23] M. Boratko *et al.*, “A systematic classification of knowledge, reasoning, and context within the arc dataset,” in *Proc. Workshop Mach. Reading Question Answering*, 2018, pp. 60–70.
[24] S. Jain, “Question answering over knowledge base using factual memory networks,” in *Proc. NAACL Student Res. Workshop*, 2016, pp. 109–115.
[25] A. Neelakantan, Q. V. Le, and I. Sutskever, “Neural programmer: Inducing latent programs with gradient descent,” in *Proc. Int. Conf. Learn. Representations*, 2016.

[26] P. Yin, Z. Lu, H. Li, and B. Kao, "Neural enquirer: Learning to query tables with natural language," in *Proc. Int. Joint Conf. Artif. Intell.*, 2016, pp. 2308–2314.
[27] F. Hill, A. Bordes, S. Chopra, and J. Weston, "The goldilocks principle: Reading children's books with explicit memory representations," 2015, *arXiv:1511.02301*.
[28] J. Weston *et al.*, "Towards ai-complete question answering: A set of prerequisite toy tasks," 2015, *arXiv:1502.05698*.
[29] N. De Cao, W. Aziz, and I. Titov, "Question answering by reasoning across documents with graph convolutional networks," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Human Lang. Technol.*, 2019, pp. 2306–2317.
[30] M. Tu, G. Wang, J. Huang, Y. Tang, X. He, and B. Zhou, "Multi-hop reading comprehension across multiple documents by reasoning over heterogeneous graphs," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 2704–2713.
[31] Y. Cao, M. Fang, and D. Tao, "BAG: Bi-directional attention entity graph convolutional network for multi-hop reasoning question answering," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Human Lang. Technol.*, 2019, pp. 357–362.
[32] D. Chen, A. Fisch, J. Weston, and A. Bordes, "Reading wikipedia to answer open-domain questions," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics*, 2017, pp. 1870–1879.
[33] M. Dunn, L. Sagun, M. Higgins, U. Guney, V. Cirik, and K. Cho, "SearchQA: A new q&a dataset augmented with context from a search engine," 2017, *arXiv: 1704.05179*.
[34] B. Dhingra, K. Mazaitis, and W. W. Cohen, "QUASAR: Datasets for question answering by search and reading," 2017, *arXiv: 1707.03904*.
[35] S. Wang *et al.*, "R3: Reinforced ranker-reader for open-domain question answering," in *Proc. Conf. Artif. Intell.*, 2018, pp. 5981–5988.
[36] C. Clark and M. Gardner, "Simple and effective multi-paragraph reading comprehension," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, 2018, pp. 845–855.
[37] S. Wang *et al.*, "Evidence aggregation for answer re-ranking in open-domain question answering," in *Proc. Int. Conf. Learn. Representations*, 2018.
[38] Y. Lin, H. Ji, Z. Liu, and M. Sun, "Denoising distantly supervised open-domain question answering," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, 2018, pp. 1736–1745.
[39] Z. Wang, J. Liu, X. Xiao, Y. Lyu, and T. Wu, "Joint training of candidate extraction and answer selection for reading comprehension," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, 2018, pp. 1715–1724.
[40] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. Int. Conf. Learn. Representations*, 2015.
[41] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2017.
[42] D. Beck, G. Haffari, and T. Cohn, "Graph-to-sequence learning using gated graph neural networks," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, 2018, pp. 273–283.
[43] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[44] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[45] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky, "The stanford corenlp natural language processing toolkit," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics: Syst. Demonstrations*, 2014, pp. 55–60.
[46] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation," in *Proc. Int. Conf. Empir. Methods Natural Lang. Process.*, 2014, pp. 1532–1543.
[47] J. Chen, S.-t. Lin, and G. Durrett, "Multi-hop question answering via reasoning chains," 2019, *arXiv: 1910.02610*.
[48] Y. Zhuang and H. Wang, "Token-level dynamic self-attention network for multi-passage reading comprehension," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 2252–2262.
[49] J. Chen and G. Durrett, "Understanding dataset design choices for multi-hop reasoning," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Human Lang. Technol.*, 2019, pp. 4026–4032.
[50] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Human Lang. Technol.*, 2019, pp. 4171–4186.

Linfeng Song received the MS degree in computer science from the Institute of Computing Technology, Chinese Academy of Science, in 2014, and the PhD degree from the Computer Science Department, University of Rochester. He is currently a senior researcher with Tencent AI Lab. His research interests including dialogue understanding, semantic parsing and question answering.

Zhiguo Wang received the PhD degree in computer science from the Institute of Automation, Chinese Academy of Science, in 2012. He joined IBM Research as a research staff member in 2014, and currently a senior applied scientist and manager at Amazon Web Service, working on question answering.

Mo Yu received the PhD degree from the Harbin Institute of Technology, in 2016. He is currently a research staff member at IBM Research.

Yue Zhang received the PhD degree from the University of Oxford, in 2009, and did a postdoc with the University of Cambridge. He was an assistant professor with the Singapore University of Technology and Design and is currently a tenured associate professor with Westlake University.

Radu Florian received the PhD degree from Johns Hopkins University, in 2002. He is currently a senior research manager with IBM Research.

Daniel Gildea received the BS, MS, and PhD degrees from the University of California at Berkeley, in 1995, 1999, and 2001, respectively. He is currently a tenured full processor with the Computer Science Department, University of Rochester.