Position-Aware Neural Attentive Graph Networks for Multi-hop Question Answering

Ege Ersü, Yifu Qiu, Anda Zhou
The University of Edinburgh
School of Informatics

Abstract

Recently there has been considerable interest in applying graph neural networks (GNN) to multi-hop question answering (QA) tasks, as graph representations can explicitly express rich dependencies in language. However, graph representations suffer from the loss of sequential information and the difficulty of representing global semantic information. In this work, we propose the *query-attention mechanism* to enhance the GNN-QA system by utilizing both global and local contextual information. We also explore injecting the positional information into the graph to complement the sequential information. Our experiments are conducted on the WikiHop dataset to allow direction comparison with Entity Relational-Graph Convolutional Networks (De Cao et al., 2019). Our contributions identify the existence of *position bias* in the dataset and our experiment results with ablation study confirms that our proposed modules improve the generalization accuracy by 1.43%.

1. Introduction

Teaching machines to reason is one of the most challenging tasks in natural language processing (NLP), which not only requires the understanding of language but also being capable to identify and extract the supporting facts to respond to a query. Most previous work in this field mainly focused on single-document reading comprehension question answering (QA): the information source for answering the question is entailed by the given document. The large-scale open-source QA datasets (Rajpurkar et al., 2016) enable the end-to-end neural models to become the state-of-the-art (Seo et al., 2017; Weissenborn et al., 2017). Recently, question answering based on multiple documents has been proposed, or Multi-hop question answering (Multi-hop QA). Compared with single-document QA, multi-hop QA is more challenging as it requires reasoning across a number of given supports documents, and the system cannot achieve promising performance by solely relying on local information in any single support.

Although Multi-hop QA is intended to evaluate the reasoning of QA system in multi-document scenarios, one of the feasible solutions is still to extend the model on single-document QA. In (Welbl et al., 2018), they reported the performances of several state-of-the-art RNN-based models including BiDAF (Seo et al., 2017) and FastQA (Weissenborn et al., 2017) by concatenating all supporting documents into one. Although the RNN-based model still beats the baselines of the rule-based model, such naive solution bears the harm in performance by its *relational inductive bias* (Battaglia et al., 2018) in its sequential structure. The RNN-based model essentially discards the parallel structure of multiple supports, and the long-term forgetting problem becomes serious when taking long sequence as input.

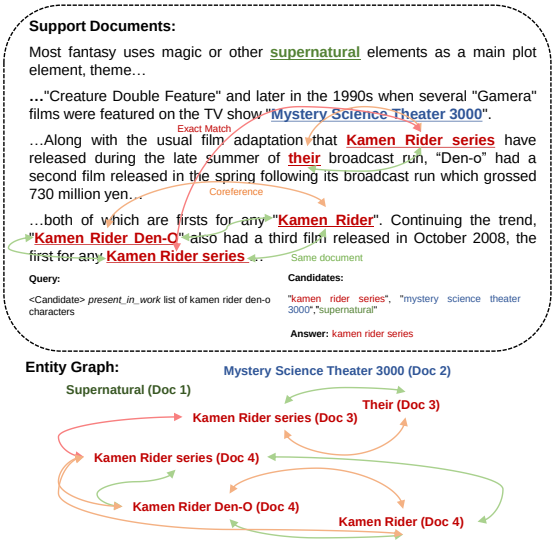

Figure 1. Top panel: Illustration for the multi-hop question answering task on WIKI_DEV_002 sample. Multi-hop reasoning requires the aggregation of information from different supports. Bottom panel: Entity Graph representation proposed in (De Cao et al., 2019). We do not show the `COMPLEMENT` edges for brevity.

With the recent rise of the graph neural network (GNN), another method being studied is to represent text in a structured form, i.e., graph, and assume the *message passing mechanism* in GNN can explicitly aggregate different supporting information over the graph (De Cao et al., 2019; Ding et al., 2019; Qiu et al., 2019; Thayaparan et al., 2019). Nevertheless, the graph-based text representation has shortcomings: compared with RNN-based text encoding, RNN's recurrent structure naturally captures the sequential property of text, but the graph representation treats each *semantic unit* (Peyrard, 2019) as parallel nodes connected by specified relationships, which essentially lose the order of

	Min	Max	Avg.	Median
# candidates (WikiHop)	2	79	19.8	14
# documents (WikiHop)	3	63	13.7	11
# tokens/doc. (WikiHop)	4	2,046	100.4	91
# candidates (ours)	2	70	20	20
# documents (ours)	3	61	13.8	12
# tokens/doc. (ours)	7	1,925	98.9	89

Table 1. Comparison of training set statistics between our splited sub-dataset (7,891 samples) and full WikiHop dataset (43,738 samples) (Welbl et al., 2018).

words in language. Additionally, RNN can not only generate the local embedding for tokens at each encoding step but also the global representation for the whole sequence in its hidden states. However, though graph pooling, e.g., average, provide some heuristics for encoding the graph, it is still difficult to encode with specific to downstream task like RNN via an fully data-driven fashion.

We argue that both shortcomings are crucial for the GNN-QA system with entity graph: 1) sequential information is important for learning contextual information, while GNN-QA system relies heavily on the entity's context to inference correct answers. 2) similar to other NLP tasks (Cohn et al., 2016; Ko et al., 2020), Multi-hop QA has the *position bias*, i.e., the position of each entity mentions contains the information to discriminate whether it is an answer, 3) global contextual representation of graph should improve the model by learning to enhance the importance of answer nodes with specific to query.

To tackle the aforementioned defects in the GNN-QA system, we propose *Position-aware Query-Attention Graph Networks* (Pos-QAGN) in this paper. Inspired by the positional embedding in Transformer (Vaswani et al., 2017), we complement the discarded sequential information in GNN by injecting the positional embedding into nodes, and compare two types of injection. A QA-specific *query-attention mechanism* is further designed to enhance the global representation. We experiment our model with the Entity Relational-Graph Convolutional Network (Entity-RGCN) in (De Cao et al., 2019) as our backbone [1] on part of WikiHop dataset (Welbl et al., 2018). In order to compare with the baseline on our sub-data set, we reproduced the Entity-RGCN model and improved the offline graph generation algorithm with more lightweight storage requirement. The experiment results show that our full Pos-QAGN model with late positional injection achieves over 1.43% improvement over the Entity-RGCN baseline model report by us. Furthermore, we both quantitatively and qualitatively analyzed the position bias and query-attention mechanism, which verified the effectiveness of the proposed modules.

We summarize our main contributions as followed,

- We reproduced and open sourced the first community version of Entity-RGCN in (De Cao et al., 2019) with an storage efficient solution, in which we reduce the storage requirement at 1TB to around 23GB [2].
- We identify the existence of *position bias* in the WikiHop dataset (Welbl et al., 2018), and further propose to complement the lost sequential information in graph representation of text by positional injection.
- We propose the *query-attention mechanism* allowing our model to incorporate both global and local contextual information during reasoning. Our empirical experiment demonstrate its positive contribution on R-GCN baseline in (De Cao et al., 2019).

2. Data set and task

Data. We conduct our experiment on the WikiHop dataset (Welbl et al., 2018). Each data sample in WikiHop is a tuple $< q, S_q, C_q, a^* >$: q is the query, consisted by a triples $< s, r, o_? >$ where s are the subject, and $o_?$ denote the unknown object entity that the QA system needs to infer. The r denotes the relation between $s, o_?$. For the details about constructing WikiHop dataset, please see in (Welbl et al., 2018). S_q is a set of supporting documents for particular query q. C_q is the set of candidate answers for the query q, which should entail the ground-truth answer a^*. We illustrate one WikiHop sample in Figure 1.

Dataset. Limited by computing resources, we split out a part of the WikiHop dataset (Welbl et al., 2018) for conducting our experiments. We first select 8,000 samples from the WikiHop training set to train our models. As there is no public testing set for WikiHop, we select 1,000 samples from the WikiHop development set and halve them into our validation set (500) and test set (500). Then we perform the same pre-processing procedure as in (De Cao et al., 2019), which removes the samples that satisfying one of the following criteria: 1) sample contains more than 500 nodes in its generated entity graph, 2) there is no node in entity graph corresponding to the answer entity. Finally, there are 7,891 training samples, 486 and 489 samples for validation and testing in our dataset. We report the comparison of several statistics between our dataset with the original WikiHop dataset on number of candidates and support documents per sample, and the document length in Table 1. It turns out our small dataset has similar characteristics as the original one, except that the distribution of candidates number per sample is more symmetrical.

Task. Following previous notations, we define our task formally: the goal for Multi-hop QA is to find the correct answer a^* for given query q from a set of candidates C_q by combining multiple facts that are spread across multiple support documents S_q. Specifically, each sample $<q, S_q, C_q, a^* >$ should satisfy that $a^* \in C_q$ and all entities in C_q are mentioned in S_q.

3. Methodology

In this section we describe 1) offline entity graph construction algorithm with positional embedding 2) GNN module for Multi-hop reasoning and 3) the query-attention mecha-

[1] Practically, the positional-aware design and query-attention mechanism can be applied on any GNN-QA system.

[2] Our code is available at: https://github.com/egeersu/Multihop-GNN

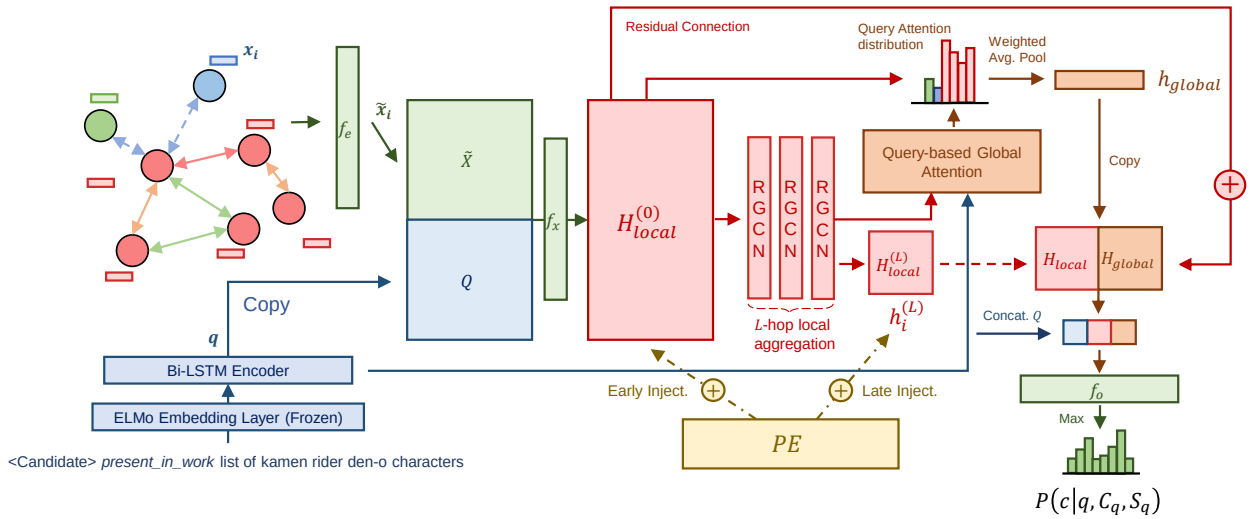

Figure 2. Model architecture for our Query-Attentive Relational-Graph Convolutional Networks.

nism which allows the model to utilize the global contextual information when predicting the answer.

3.1. Entity Graph Construction

Similar to (De Cao et al., 2019), we use the entity graph formalism to organize and represent the content of support documents. The nodes of the entity graph consist of all entity mention spans in support documents where the mentions should belong to the set of entities in candidate answers C_q or subject $\{s\}$ of the query. There are several heuristics to extract the mentions as a node. Firstly, we identify the mentions spans in the support document which exactly matches with the element in the entity set $C_q \bigcup \{s\}$. Then, we use the neural coreference system (Lee et al., 2017) to add the referent mentions as the nodes. This step would expand the node set with their noun phrases and pronouns. As the case is shown in Figure 1, the coreference predictor successfully identifies "Kamen Rider" as a referent of "Kamen Rider Den-O", where the latter might be the key to identify the answer as its "Den-O" token also occurs in the query. In the end, we discarded the mention spans which have multiple coreferent chain as suggested in (De Cao et al., 2019), because it would help the GNN module to avoid ambiguous propagation of information.

We use the pre-trained ELMo word embedding in (Peters et al., 2018) as the contextualized node representations. Each node is assigned with a continuous representation by averaging over the token embeddings in the span. For the sake of simplicity, we do not fine-tune parameters for ELMo embedder in the later online training step. In addition to baseline setting, we compute the positional embedding (Vaswani et al., 2017) for each node span based on the location of its first token,

$$\begin{aligned} PE_{(start,i,2k)} &= sin(start/10000^{2k/d_{model}}) \\ PE_{(start,i,2k+1)} &= cos(start/10000^{2k/d_{model}}) \end{aligned} \quad (1)$$

where $start$ is the start position of i node and k is the dimension. d_{model} is the dimension for positional embedding. We will compare two injections of positional information: i) Early injection: we add the positional embedding to node representations before the message passing in GNN module, which means we allow the positional information to participate the local information propagation, ii) Late injection: we inject the positional embedding on the node representation after message propagation, which will only affect the global representation and final predictions.

After initializing the nodes, we start to connect the node pairs by relational edges. As in (De Cao et al., 2019), there are four types of relations R: 1) `COREF`: if two nodes are predicted in the same coreferent chain (Lee et al., 2017). 2) `MATCH`: if two node spans are exactly matched with each other, 3) `DOC-BASED`: if two nodes are in the same support document, 4) `COMPLEMENT`: we add this relation to all node pairs that did not belong to any of others. The `COMPLEMENT` edges is important to prevent disconnected graph by complementing the graph to a fully-connected topology.

3.2. Multi-hop Reasoning by Relational-Graph Convolutional Network

The core idea of graph neural network (GNN) is to propagate the information contained by each node to its neighbors in each layer, through a differentiable message passing algorithm. By stacking L share-parameter GNN layers, each node in the graph can receive the information passed from its L-hop neighbor nodes, then update its node embedding. We refer such information propagation as *local information aggregation*, as each node can only communicate with neighbors in limited distance L.

In our task, we set entity mentions from different documents and contexts as nodes. Remember the core challenge for multi-hop QA is the system needs to identify and integrate the supporting facts from several documents to generate the answer. Therefore, by adopting the message passing mechanism in GNN, we can naturally update the node representations by aggregating the information in other contexts from its neighbor nodes.

Similar to our baseline (De Cao et al., 2019), we utilize the

relational-graph convolutional neural networks (R-GCN) (Schlichtkrull et al., 2018) as the backbone for message passing. R-GCN is an extension version of the Graph Convolutional Networks (Kipf & Welling, 2017), by modeling different types of edge with separate parameterized weights. The update of node information in each layer could be formulated as,

$$h_i^{(l+1)} = \sigma(W_0^{(l)} h_i^{(l)} + \sum_{r \in R} \sum_{j \in N_i^r} \frac{1}{c_{i,r}} W_r^{(l)} h_j^{(l)}) \tag{2}$$

where N_i^r is the set of neighbours of node i with relation r. h_i^l is the hidden vector for node i in layer l. R contains all relation types, i.e., {`COREFERENCE`, `EXACT MATCH`, `DOC-BASED`, `COMPLEMENT`}. $W_0^{(l)}, W_r^{(l)}$ are the weights for node vector in layer l with respect to particular relation r. σ is the non-linear activation. $\frac{1}{c_{i,r}}$ is the normalization factor which controls how much information are aggregated from neighbors. $c_{i,r}$ is defined by the in-degree for node i in the subgraph with single relation r.

3.3. Query-Attention Mechanism

The message passing in GNN provides an efficient way to aggregate information from neighbors for each node. However, we argue the global information of the entity graph is also very important for predicting the answer. A query-attention mechanism is proposed to learn global contextual representation for the entity graph depending on the query.

Concretely, we let the query embedding q to attend every contextualized node embedding after L-hop local propagation, then compute the attention distribution over all nodes. To be noticed, each data sample is converted to an entity graph with a fixed max node number, thus we adopt a *node mask* to filter out all the padding nodes. By doing so we can force the attention mechanism to only focus on the nodes containing meaningful entity mentions. The calculation for query-attention weight for each node i is formulated by,

$$score(q, h_i^{(L)}) = q^\top W_a h_i^{(L)} \tag{3}$$

$$a_{global,i} = \frac{exp(score(q, h_i^{(L)}))}{\sum_{i'}^{|V|} exp(score(q, h_{i'}^{(L)}))} \tag{4}$$

where W_a is the weights for aligning the dimension of query and node vectors, and V is the set of nodes. Then the global embedding for entity graph could be calculated by averaging the node embedding with attention weight $a_{global,i}$,

$$h_{global} = \frac{1}{|V|} \sum a_{global,i} h_i^{(L)} \tag{5}$$

Intuitively, the global graph representation enables our model to predict the answer distribution based on both the local information from neighbors $h_i^{(L)}$, and the global information h_{global} of the entire graph. The global representation is computed according to the importance of different nodes by attending them with the query. Therefore, query-attention provides a learnable way for our model to enhancing the representations of those nodes having high relativeness with the query.

3.4. Model Architecture

Our model is similar to that of (De Cao et al., 2019), and is depicted in Figure 2, except that we remove some hidden layers to simplify the model and suit our smaller dataset. Firstly, a single-layer feed-forward network is adopted to reduce the dimension of ELMo embedding for each node, i.e., $\tilde{x}_i = f_e(x_i)$. Next, we embed the query by a frozen ELMo embedding layer and apply an extra trainable query encoder (a single-layer bidirectional LSTM taking the ELMo vector of each token as input) to generate the query embedding q by concatenating the forward and backward hidden states. Then we concatenate the query with each node embedding and feed them into a single-layer feed-forward network f_x to form the initial node representation $h_i^{(0)}$ and start the local message passing via a stacked-layer R-GCN as in Eqt. (2),

$$h_{local,i}^{(L)} = R\text{-}GCN_L(\overbrace{f_x([\tilde{x}_i, q])}^{h_{local,i}^{(0)}}, N_i^R, C_i^R) \tag{6}$$

where $h_{local,i}^{(L)}$ is i node's local representation after L layers R-GCN propagation. N_i^R is the neighbours set for node i specified by the relation r in all relation types R. C_i^R stores the in-degree information for node i for each relation r in R. To prevent the gradient problem in training, a residual connection (He et al., 2016) is employed starting from the input layer of R-GCN to the output of query-attention. That is, a global and local-aware node representation after concatenating the outputs of R-GCN and query-attention is,

$$\tilde{h}_i = [h_{local,i}^{(L)}, h_{global}] + W_{res} h_{local,i}^{(0)} \tag{7}$$

where W_{res} is the linear projection for matching dimensions. Finally, we concatenate $\tilde{h}_i$ with the query vector for each node and fed into a final feed-forward networks f_o for generating the distribution over candidates and we train the model to maximize the likelihood of observations,

$$P(c|q, C_q, S_q) \propto exp(\max_{i \in \mathcal{M}_c} f_o([q, \tilde{h}_i])) \tag{8}$$

where $\mathcal{M}_c$ is the set of nodes whose mention is candidate c. The `max` operation is to select out the highest prediction probability node from all of that candidate entity.

4. Experiments

We introduce our experiments in this section. Firstly, two major experiments including position bias evaluation and comparison experiment are introduced. We also report the estimated time and computing resources required during the experiment, because we believe this will be helpful for other future works.

4.1. Position Bias Evaluation

Compared to our backbone in (De Cao et al., 2019), PosQAGN additionally applies the positional embedding on the entity mention for each node. We assume that the position information can help the Multi-hop QA system to determine whether an entity is an answer or not. To further support our assumption, we first visualize the positional distribution of the answer entities and other entities in Figure 3, which counts the relative frequency of entity mention

Model Types	Model	Train NLL	Train Acc.	Val. NLL	Val. Acc.
Rule	Random	-	11.75	-	10.75
	Max-Mention	-	11.90	-	11.00
	Majority-Candidate-per-Query-Type	-	**58.17**	-	28.00
Sequence	BiDAF (Seo et al., 2017)	2.08	34.91	2.11	28.12
Graph	Entity-RGCN (De Cao et al., 2019)	1.82	44.84	2.09	35.80
	Pos-QAGN (Early)	1.72	48.74 (↑)	2.10	36.63 (↑)
	Pos-QAGN (Late)	1.68	50.29 (↑)	2.09	**39.30** (↑)

Table 2. Performance comparison for different models in three categories (rule-based models, sequence neural model and graph neural models) in our training and validation set. We report the metrices in both loss and accuracy.

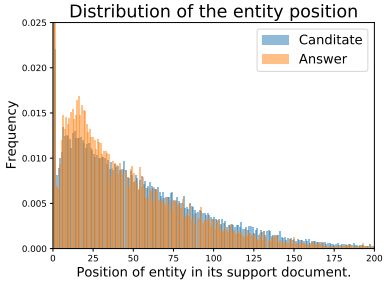

Figure 3. Distribution of entity position in the support document

Epoch	Train					Val.				
	Acc.	Loss	P.	R.	F1.	Acc.	Loss	P.	R.	F1.
1	89.27%	108.6	0.84	0.55	0.661	45.8%	43.31	1	0.5	0.667
10	90.2%	15.89	1	0.5	0.667	45.8%	21.22	1	0.5	0.667

Table 3. Training result for the position bias binary classifier on both training and validation sets.

appearing in various positions of the support document. It can help us to identify whether a candidate is answer is related to its position in the document. If there is a difference for positional distributions between answer and non-answer candidates, simply using position information might distinguish whether an entity is an answer and surpass the coin-flipping baseline.

To further quantify how much information about answer in the position of the entity, we train a binary classifier that takes the position embedding of the candidate entity mentions as input, and predicts whether the candidate is the answer. The experiment is conducted on our small dataset with a three-layer feed forward networks. The dimensions are set to be 512, 128, 64, respectively. We use the cross-entropy as the loss and train the classifier with maximum likelihood estimation. The result is reported in Table 3.

4.2. Comparison Experiment

Our major contribution is landing on the improvement of model performance by two proposed modules: positional injection and query-attention mechanism, we will dominantly focus on the comparison with baselines and ablation study on each individual module in our experiment. We also further compare patterns in the attention distribution for three attentive models in our experiments, and analyze how the query-attention works.

Hyper-parameters. For two versions of Pos-QAGN, our model has 256-dimensional hidden states for node dimension reduction and query embedding, and 512-dimensional node representations during message propagation. To match the dimension with the R-GCN module, we also use 512-dimensional hidden states for positional embedding and query attention. For the last feed-forward networks for outputting the answer distribution, the dimensions are 1024 and 256, respectively.

Training Details. As mentioned, we do not fine-tune the ELMo layers during our online training. We use all three-layer hidden states from ELMo embedder by concatenating them together. Following (De Cao et al., 2019), we set the hop number of R-GCN to be 3. We train the models in mini-batch with size at 32 by ADAM (Kingma & Ba, 2014). The learning rate is set to be 1×10^{-4}. As our dataset is much smaller than the original WikiHop, we adopt dropout (Srivastava et al., 2014) after each layer with a drop rate of 0.25. Finally, we train all models with 10 epochs as maximum and employ validation loss as the criterion with 3 epochs patience to implement early stopping.

Baseline. To quantify the improvement brought by our proposed module, we select the backbone model we based on as our major baseline (De Cao et al., 2019). We also reproduce and test four baselines as reported in (Welbl et al., 2018) to confirm the advantage of the graph-based models also existing in our sub-dataset:

i) *Random*: We start by picking a random candidate for each sample. This gives us the minimum accuracy that any model should surpass.

ii) *Max-Mention*: For each sample, the candidate that occurs most frequently in the support documents is picked. If there is a tie, we randomize over the remaining candidates. The result is almost identical to the Random baseline, indicating that the dataset was constructed in a way that does not show any bias for frequency.

iii) *Majority-Candidate-per-Query-Type*: In training, this model simply counts the correct answers for each query (e.g., *place_of_birth*), then it simply output the candidate that has occurred the most with that query in testing. This baseline achieves considerable success when testing (29.20%), indicating that certain candidates indeed happen more frequently with certain query types.

iv) *BiDAF*: To compare GNN-based model to a strong LSTM-based QA architecture, we decided to report the Bi-directional Attention Flow (BiDAF) as an end-to-end alternative (Seo et al., 2017). The model was initially released to predict an answer span on a given document, working

Model Description	Model	Acc.	Top-2	Top-5
Baselines in (Welbl et al., 2018)	Random	11.27	17.6	18.4
	Max-Mention	13.00	22.00	22.60
	Majority-Candidate-per-Query-Type	29.20	34.20	34.60
	BiDAF	29.42	39.28	56.03
Baseline in (De Cao et al., 2019)	Entity-RGCN	37.63	52.15	**73.42**
Ours	Pos-QAGN (Early)	**39.06** (↑)	**52.76** (↑)	72.60
	Pos-QAGN (Late)	37.63	51.33	72.19

Table 4. Performance comparison for different models in our splited testing set in terms of accuracy (Acc.), Top-2 and Top-5 precision.

robustly on QA tasks like SQuAD where the answer has to be a segment of text (Rajpurkar et al., 2016). In order to train the model on multi-document QA task, we follow the method proposed by (Welbl et al., 2018) to concatenate all support documents into one large sequence as the input of this model. To suit with our smaller dataset, we downscale the default parameters and set the hidden size down to 15, the batchsize to 16 and train for 2,500 iterations.

Our comparison results in training and validation are shown in Table 2, and the testing are reported in Table 4. We evaluate our models by the accuracy and negative log-likelihood loss (NLL) for training and validation, and top-K precision with $K = 1, 2, 5$ which counts correct if ground truth are in the K highest predictions in testing.

Abalation and Case Study. To further determine the sources of improvements, we conduct the ablation study for our two full models (Pos-QAGN Early and Late version) on validation set (see Table 5) by re-runing the models without one or both of query-attention and positional injection. To better understand the mechanism of query-attention. We randomly picked 4 samples in the validation set from the correct predictions given by all three models equipped with the query-attention, and further visualize the attention distribution over candidates in Figure 4. We exclude the attention weights that not belong to the candidates and sum all the weights for each candidate.

4.3. Experiment Details

We started our experiments with the original dataset of WikiHop, and we found that graph generation has a huge storage overhead. According to our estimates, it will take about 432 hours to run graph generation for the complete dataset on an Intel(R) Xeon(R) E5-2650 @ 2.2GHz CPU. At the same time, the generated graph with padding will take up about 1034GB of storage. Therefore, we use the graph structure in the *Deep Graph Library* (Wang et al., 2019), which stores the graphs with a serialized version for saving storage. Furthermore, we set the training process to load data only when it needs on a batch basis. By doing so, the storage of all entity graphs could be reduced to around 23 GB. However, this would be more time-consuming during training, as some operations such as padding need to be repeated per epoch. We estimate that training on the full dataset will take approximately 120 hours on one NVIDIA TITAN X GPU for 10 epochs. The training of graph-based models is also memory-hungry. When the batch size is set to 32, the peak memory overhead of model training will reach 104GB. Therefore, considering both of time and computing resources, we will conduct experiments on the sub-dataset as described in previous section.

Description	Model	Train Acc.	Val. Acc.
Full Model	Full (Early PE Inject.)	48.74	36.63
	Full (Late PE Inject.)	50.29	39.30
Components	w/o Query-Attention (Early)	48.49 (↓0.25)	38.27 (↑1.64)
	w/o Query-Attention (Late)	49.20 (↓1.09)	37.86 (↓1.44)
	w/o PE Injection	44.85 (↓3.89, ↓5.44)	37.65 (↑1.02, ↓1.65)

Table 5. Ablation study on query-attention mechanism and positional injection in Pos-QAGN model.

5. Results and Discussion

5.1. Position Bias

Obviously, in Figure 3, the answer entities are more concentrated in the front position of each document than non-answer entities, which means the position information might help the QA system to identify whether a candidate is an answer.

The quantitative result predicted by our binary classifier in Table 3 shows that the predicted accuracy of whether a candidate mention is an answer or not is over 90%. However, as the number of answer entities and non-answer entities is highly unbalanced (around 1:10), this high accuracy might be not so meaningful. On the other hand, the 0.55 recall can convince us that there are more than half of the answer entity can be predicted correctly. The 0.66 harmonic mean of precision and recall also surpass the 50% baseline of coin-flipping prediction.

In a nutshell, the location of the entity does imply the position bias of whether the entity is an answer in the Multi-hop QA dataset of our experiment. Therefore, it might be possible to improve the overall accuracy by injecting the positional information into the GNN-QA system to complement the positional information of the entity mention in each node.

5.2. Comparison with Baseline

Given that we compare our model in the sub-set of WikiHop, we first look at the comparison between our Entity-RGCN baseline with the sequence model (BiDAF). Table 4 shows that the Entity-RGCN still outperforms than 4 baselines reported in (Welbl et al., 2018), including both rule-based and RNN-based neural model. This is consistent with the comparison result in (Welbl et al., 2018). It demonstrates that regardless of the size of the dataset, graph neural

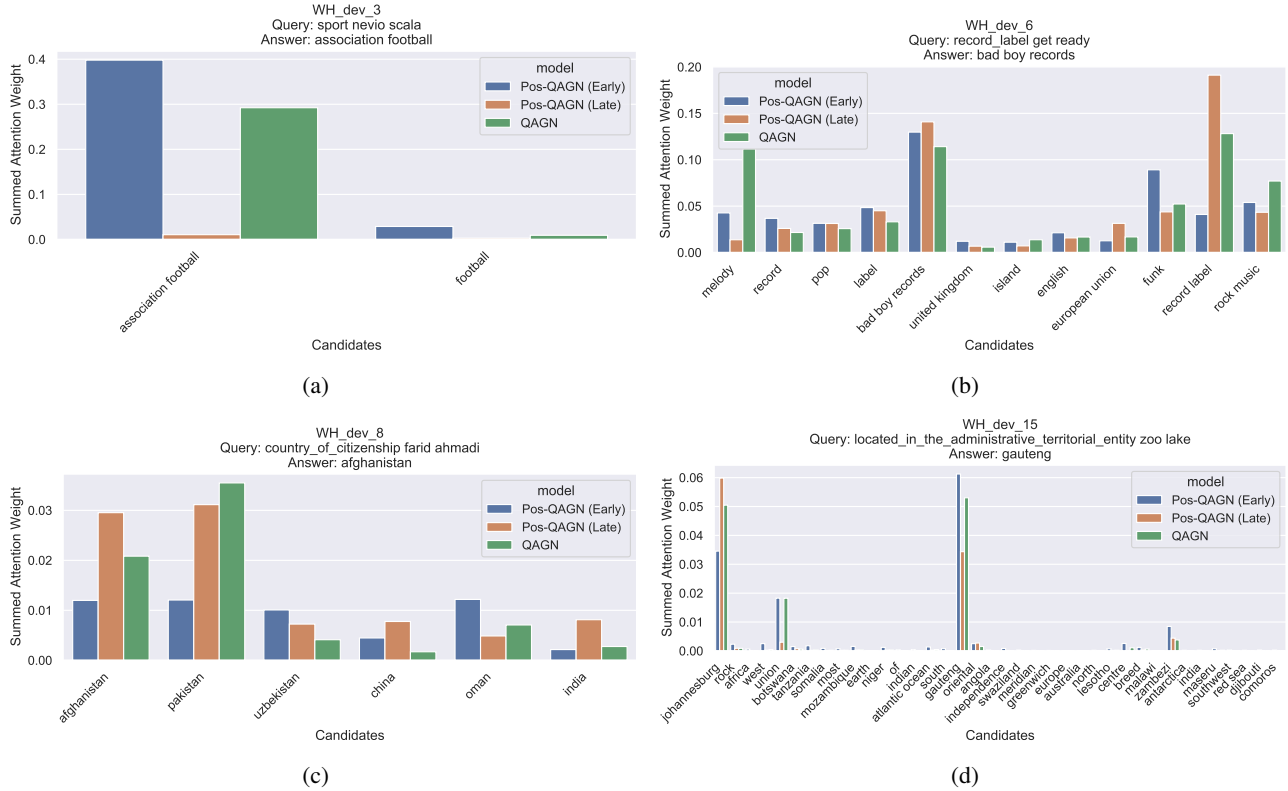

Figure 4. Visualization for the query-attention distribution of QAGN in four picked data sample with correct predictions: WH_dev_3, WH_dev_6, WH_dev_8 and WH_dev_15.

networks have advantages than the RNN-based model in Multi-hop QA tasks: models based on graph neural networks can achieve lower loss on the training set, and have better generalization accuracy with a considerable margin.

Subsequently, we compare our two Pos-QAGN with Entity-RGCN. Table 2 reveals that the model equipped with our two proposed modules can contribute to the baseline to achieve lower training loss, where the late and early injection of positional embedding reduces the training loss by 0.14 and 0.1, respectively. This shows that our model can have a stronger capability to fit the dataset. We also observe the improvement in validation accuracy of our Pos-QAGN compare to baseline by 0.83% and 2.9%.

Nevertheless, in testing, only the early injection of positional embedding (Pos-QAGN with Early Injection) still outperforms Entity-RGCN in accuracy, with improvement at around 1.5%. The late injection Pos-QAGN has the same prediction accuracy as the baseline. It might because our test set of 489 samples is not large enough to estimate the disparity of generalization performance between Pos-QAGN (Late) and the baseline. And we have also observed that with the relaxation of precision calculation (K increases), the improvement of our model gradually decreases, and even worse performance is achieved when calculating Top-5 precision. The Pos-QAGN (Late) model has lower Top-2 and Top-5 precision compare to baseline, and Pos-QAGN (Early) only has the advantage in Top-2 precision with 0.61 improvement, which is much smaller than the improvement in accuracy. It might because query-attention can make the prediction distribution to be spike, which forces the output prediction networks to more focus on a part of nodes in the graph, and we will discuss this later.

5.3. Ablation Study

Positional Information. In Table 5, we observe that compare with our two full models, the model without positional injection (QAGN) has considerable performance drops in the training set. But in validation, removing positional information decrease the accuracy of the late-inject full model while increasing the accuracy of our early-inject full model.

Query-attention Mechanism. We do perform one last ablation to remove the query-attention mechanism, which forces the model to only look at local contextualized information for each node. As expected, the training performance drops considerably by removing the query-attention mechanism, with gap at 1.64 and 1.44 for two full models. However, we again observe that there is an increase of validation accuracy by removing the early-inject full model.

The possible reason for this in-consistent observation might be either the full model overfit the training set, as we did not fine-tune hyperparameters for every single model but majorly following the setting in (De Cao et al., 2019), or the small size of validation set results in a noisy performance estimation. We remain finding this answer as the future work.

5.4. How Does Query Attention Work?

In general, we found similar patterns in the attention distributions of three attentive models. Query-attention tends to give high weights to a small number of nodes, thereby enhancing their saliency in the global representation. Subsequently, we found that most of the nodes with high attention weight are the answer nodes, or have a strong relationship with the query. For example, in the WH_dev_6, answer nodes *bad boy records* and the *record label* having the same token in the query are assigned with the highest weight. We also find that the query attention distribution is quite spiking and sharp, as the case in Figure 4(d), *ohannesburg* and *gauten* account for more than 90% over the weight. This may cause the predictive distribution $P(c|q, C_q, S_q)$ to be more centralized to small group of candidates, which makes the query-attention not very robust that if the answer node is not assigned with the highest weight, query attention may confuse the prediction of the QA system. Given that, it is no surprise to see the model has less advantage in Top-K precision if K goes larger. Furthermore, it might also cause the model easier to overfit, which requires a finer hyperparameter search to maximize the performance.

5.5. Early Injection vs. Late Injection

We have shown that the query-attention mechanism enhances the node representation related to the query in the global representation, and the positional information is useful in WikiHop dataset to find the correct answer. But, *either early or late positional injection, which is better?*

Figure 4 shows the comparison in attention distribution grouped by different types of injection. It turns out early injection tends to learn a "softer" attention distribution, i.e., the weights are more scattered. The attention weights learned by the late injection model are more centralized to a particular few candidate nodes, and unfortunately, which might be not related to the ground truth answer (e.g., *johannesburg* in WH_dev_15). This means that once the attention distribution cannot correctly reflect the structural bias between the answer and other candidates, it will be harder for the late-injection model to correct the node preference in the global representation. It might be one explanation why we see that Pos-QAGN Late has more degradation of model performance on the test set than the early one.

Another interesting observation is the position embedding plays an important role in RGCN's message propagation and helps the query-attention to ignore distractions in some cases. In Figure 4(b) and (c), all attentive models give the answer node and a non-answer candidate node with considerable higher weight than other candidates, but for QAGN and late inject model which do not involve positional information to local message passing, they pays more attention to the non-answer node (distractor). On the other hand, the early inject model that brings the location information into the local message propagation effectively reduces the weight of the distractor and ensures that the answer node has the highest attention weight.

To conclude, the early injection of positional information will help the GNN module to better capture the local information, resulting in the global representation more accurately reflecting the importance of answer nodes. However, in terms of accuracy, it might worth further studying that whether late injection can surpass the early one by other methods to avoid overfitting.

6. Related work

We start by identifying a set of baseline models that would require manageable amounts of computing and memory on multi-document question answering tasks. The simplest baseline is FastQA (Weissenborn et al., 2017), an RNN-based architecture requiring minimal compute. A more complex baseline is BiDAF (Seo et al., 2017), a hierarchical multi-stage architecture that models the representations of a sequence at different levels of granularity. It utilizes character-level, word-level, and contextual embeddings in combination with bi-directional attention flow.

Other than entity graphs, there have been other attempts to use explicit structures to reasoning (Ding et al., 2019; Thayaparan et al., 2019; Qiu et al., 2019). One successful method extracts explicit reasoning chains from documents, which are simply sequences of sentences leading to the answer (Chen et al., 2019). These chains are then fed into a BERT-based architecture to predict the answer.

Despite the promising future of graph neural networks and other models that utilize explicit representations, transformer-based models (Zaheer et al., 2020) are currently state-of-the-art on WikiHop. These models still consider all support documents as long sequence. To overcome the inability to process long sequences of text of Transformer (Vaswani et al., 2017), one specific model achieved considerable success is the Longformer (Beltagy et al., 2020), which uses a modified attention operation that combines windowed local-context self-attention with an end task motivated global attention; which scales linearly instead of quadratically with sequence length.

A promising class of models called Graph Transformers was also shown to be effective in natural language generation tasks conditioned on graph structures such as knowledge bases (Koncel-Kedziorski et al., 2019). We theorize that utilizing these models in conjunction with entity-graphs might utilize the best of both worlds.

7. Conclusions

In this work, we proposed the positional injection and query-attention global representation to improve the GNN-based multihop QA system. Due to the limitation in time and computing resources, we evaluate and compare our model with baselines including Entity-RGCN in (De Cao et al., 2019) on part of WikiHop dataset. The result confirms that our proposed modules in Pos-QAGN contribute to the testing accuracy with around 1.5% improvements. The ablation and case study also further verify the effectiveness

of each individual component and their collaboration. The future work includes testing our model on more large-scale dataset and apply two proposed modules on other GNN-QA models to further verify the generalization and adaptability.

References

Battaglia, Peter W, Hamrick, Jessica B, Bapst, Victor, Sanchez-Gonzalez, Alvaro, Zambaldi, Vinicius, Malinowski, Mateusz, Tacchetti, Andrea, Raposo, David, Santoro, Adam, Faulkner, Ryan, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Beltagy, Iz, Peters, Matthew E., and Cohan, Arman. Longformer: The long-document transformer, 2020.

Chen, Jifan, ting Lin, Shih, and Durrett, Greg. Multi-hop question answering via reasoning chains, 2019.

Cohn, Trevor, Hoang, Cong Duy Vu, Vymolova, Ekaterina, Yao, Kaisheng, Dyer, Chris, and Haffari, Gholamreza. Incorporating structural alignment biases into an attentional neural translation model. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 876–885, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1102. URL https://www.aclweb.org/anthology/N16-1102.

De Cao, Nicola, Aziz, Wilker, and Titov, Ivan. Question answering by reasoning across documents with graph convolutional networks. In *Proceedings of NAACL-HLT*, pp. 2306–2317, 2019.

Ding, Ming, Zhou, Chang, Chen, Qibin, Yang, Hongxia, and Tang, Jie. Cognitive graph for multi-hop reading comprehension at scale. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2694–2703, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1259. URL https://www.aclweb.org/anthology/P19-1259.

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

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

Kipf, Thomas N. and Welling, Max. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Ko, Miyoung, Lee, Jinhyuk, Kim, Hyunjae, Kim, Gangwoo, and Kang, Jaewoo. Look at the first sentence: Position bias in question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1109–1121, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.84. URL https://www.aclweb.org/anthology/2020.emnlp-main.84.

Koncel-Kedziorski, Rik, Bekal, Dhanush, Luan, Yi, Lapata, Mirella, and Hajishirzi, Hannaneh. Text generation from knowledge graphs with graph transformers. *CoRR*, abs/1904.02342, 2019. URL http://arxiv.org/abs/1904.02342.

Lee, Kenton, He, Luheng, Lewis, Mike, and Zettlemoyer, Luke. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 188–197, 2017.

Peters, Matthew, Neumann, Mark, Iyyer, Mohit, Gardner, Matt, Clark, Christopher, Lee, Kenton, and Zettlemoyer, Luke. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL https://www.aclweb.org/anthology/N18-1202.

Peyrard, Maxime. A simple theoretical model of importance for summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1059–1073, 2019.

Qiu, Lin, Xiao, Yunxuan, Qu, Yanru, Zhou, Hao, Li, Lei, Zhang, Weinan, and Yu, Yong. Dynamically fused graph network for multi-hop reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 6140–6150, 2019.

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

Schlichtkrull, Michael, Kipf, Thomas N, Bloem, Peter, Van Den Berg, Rianne, Titov, Ivan, and Welling, Max. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pp. 593–607. Springer, 2018.

Seo, Min Joon, Kembhavi, Aniruddha, Farhadi, Ali, and Hajishirzi, Hannaneh. Bidirectional attention flow for machine comprehension. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=HJ0UKP9ge.

Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout:

A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15 (56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

Thayaparan, Mokanarangan, Valentino, Marco, Schlegel, Viktor, and Freitas, André. Identifying supporting facts for multi-hop question answering with document graph networks. In *Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-13)*, pp. 42–51, Hong Kong, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5306. URL https://www.aclweb.org/anthology/D19-5306.

Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N, Kaiser, Łukasz, and Polosukhin, Illia. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 6000–6010, 2017.

Wang, Minjie, Yu, Lingfan, Zheng, Da, Gan, Quan, Gai, Yu, Ye, Zihao, Li, Mufei, Zhou, Jinjing, Huang, Qi, Ma, Chao, et al. Deep graph library: Towards efficient and scalable deep learning on graphs. 2019.

Weissenborn, Dirk, Wiese, Georg, and Seiffe, Laura. Making neural qa as simple as possible but not simpler. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pp. 271–280, 2017.

Welbl, Johannes, Stenetorp, Pontus, and Riedel, Sebastian. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*, 6:287–302, 2018.

Zaheer, Manzil, Guruganesh, Guru, Dubey, Kumar Avinava, Ainslie, Joshua, Alberti, Chris, Ontanon, Santiago, Pham, Philip, Ravula, Anirudh, Wang, Qifan, Yang, Li, and Ahmed, Amr. Big bird: Transformers for longer sequences. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 17283–17297. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/c8512d142a2d849725f31a9a7a361ab9-Paper.pdf.