Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 5400–5407
Marseille, 11–16 May 2020

Propagate-Selector: Detecting Supporting Sentences for Question Answering via Graph Neural Networks

Seunghyun Yoon[1,2], Franck Dernoncourt[2], Doo Soon Kim[2], Trung Bui[2], Kyomin Jung[1]
[1]Department of Electrical and Computer Engineering, Seoul National University, Seoul, Korea
[2]Adobe Research, San Jose, CA, USA
{mysmilesh, kjung}@snu.ac.kr, {franck.dernoncourt, dkim, bui}@adobe.com

Abstract

In this study, we propose a novel graph neural network called propagate-selector (PS), which propagates information over sentences to understand information that cannot be inferred when considering sentences in isolation. First, we design a graph structure in which each node represents an individual sentence, and some pairs of nodes are selectively connected based on the text structure. Then, we develop an iterative attentive aggregation and a skip-combine method in which a node interacts with its neighborhood nodes to accumulate the necessary information. To evaluate the performance of the proposed approaches, we conduct experiments with the standard HotpotQA dataset. The empirical results demonstrate the superiority of our proposed approach, which obtains the best performances, compared to the widely used answer-selection models that do not consider the intersentential relationship.

Keywords: question answering, supporting sentence, graph neural network

1. Introduction

Understanding texts and being able to answer a question posed by a human is a long-standing goal in the artificial intelligence field. With the rapid advancement of neural network-based models and the availability of large-scale datasets, such as SQuAD (Rajpurkar et al., 2016) and TriviaQA (Joshi et al., 2017), researchers have begun to concentrate on building automatic question-answering (QA) systems. One example of such a system is the machine-reading question-answering (MRQA) model, which provides answers to questions from given passages (Seo et al., 2016; Xiong et al., 2016; Wang et al., 2017; Shen et al., 2017b).

Recently, research has revealed that most questions in existing MRQA datasets do not require reasoning across sentences in the given context (passage); instead, they can be answered by looking at only a single sentence (Weissenborn et al., 2017). Using this characteristic, a simple model can achieve performance competitive with that of a sophisticated model. However, in most real scenarios of QA applications, more than one sentence should be utilized to extract a correct answer.

To alleviate this limitation of previous datasets, another type of dataset was developed in which answering the question requires reasoning over multiple sentences in the given passages (Yang et al., 2018; Welbl et al., 2018). Figure 1 shows an example of a recently released dataset, the HotpotQA. This dataset consists of not only question-answer pairs with context passages but also *supporting sentence* information for answering the question annotated by a human.

In this study, we build a model that exploits the relational information among sentences in passages to classify the *supporting sentences* that contain the essential information for answering the question. To this end, we propose a novel graph neural network model named **propagate-selector (PS)**, which can be directly employed as a subsystem in the QA pipeline. First, we design a graph structure to hold information in the HotpotQA dataset by assigning each sentence to an independent graph node. Then, we connect the undirected edges between nodes using a proposed graph topology (see the discussion in the 4.2.). Next, we allow **PS** to propagate information between the nodes through iterative hops to perform reasoning across the given sentences. Through the propagation process, the model learns to understand information that cannot be inferred when considering sentences in isolation.

Passage 1, 2015 Diamond Head Classic:

① The 2015 Diamond Head Classic was a mid-season eight-team college basketball tournament... ② It was the seventh annual Diamond Head Classic tournament ... ③ No. 3-ranked Oklahoma defeated Harvard to win the tournament championship... **④** *Buddy Hield was named the tournament's MVP.*

⋮

Passage N, Buddy Hield:

① *Chavano Rainier "Buddy" Hield is a Bahamian professional basketball player for the Sacramento Kings of the NBA... ② ...*

Question: Which team does the player named 2015 Diamond Head Classic's MVP play for?

Supporting Sentences: **1-④, N-①**

Figure 1: An example of dataset. Detecting *supporting sentences* is an essential step being able to answer the question.

Unlike the previous studies, this work does not use the exact "answer span" information while detecting the supporting sentences. It shows a different way of using the HotPotQA dataset and provides researchers new opportunities to develop a subsystem that is integrated into the full-QA systems (i.e., MRQA). Through experiments, we demonstrate that compared with the widely used answer-selection models (Wang and Jiang, 2016; Bian et al., 2017; Shen et al., 2017a; Tran et al., 2018; Yoon et al., 2019), the proposed method achieves better performance when classifying *supporting sentences*.

2. Related Work

Previous researchers have also investigated neural network-based models for MRQA. One line of inquiry employs an attention mechanism between tokens in the question and passage to compute the answer span from the given text (Seo et al., 2016; Wang et al., 2017). As the task scope was extended from specific- to open-domain QA, several models have been proposed to select a relevant paragraph from the text to predict the answer span (Wang et al., 2018; Clark and Gardner, 2018). However, none of these methods have addressed reasoning over multiple sentences.

To understand the relational patterns in the dataset, researchers have also proposed graph neural network algorithms. (Kipf and Welling, 2017) proposed a graph convolutional network to classify graph-structured data. This model was further investigated for applications involving large-scale graphs (Hamilton et al., 2017), for the effectiveness of aggregating and combining graph nodes by employing an attention mechanism (Veličković et al., 2018), and for adopting recurrent node updates (Palm et al., 2018). These methods successfully demonstrated their potential and effectiveness in understanding relational datasets, such as entity linking in heterogeneous knowledge graphs, product recommendation systems, and detecting side effects in drug (Wu et al., 2019; Fan et al., 2019; Zitnik et al., 2018). In addition, one trial involved applying graph neural networks to QA tasks; however, this usage was limited to the entity level rather than sentence-level understanding (De Cao et al., 2019).

3. Task and Dataset

The specific problem we aim to tackle in this study is to classify *supporting sentences* in the MRQA task. We consider the target dataset HotpotQA, by (Yang et al., 2018), which comprises tuples ($<Q, P_n, Y_i, A>$) in which Q is the question, P_n is the set of passages as the given context, and each passage $P \in P_n$ further comprises a set of sentences S_i ($S_i \in P_n$). Here, Y_i is a binary label indicating whether S_i contains the information required to answer the question, and A is the answer. In particular, we call a sentence, $S_s \in S_i$, a *supporting sentence* when Y_s is *true*. Figure 1 shows an example of the HotpotQA dataset.

In this study, we do not use the answer information from the dataset; we use only the subsequent tuples $<Q, P_n, Y_i>$ when classifying *supporting sentences*. We believe that this subproblem plays an important role in building a full QA pipeline because the proposed models for this task will be combined with other MRQA models in an end-to-end training process.

4. Methodology

Our objective in this study is to identify *supporting sentences*, among sentences in the given text that contain information essential for answering the question. To tackle this problem, we first introduce answer-selection models, which are widely studied in the research community. These models are considered strong baselines since they can be directly applied to our task with the same objective function. Then we describe our proposed method.

4.1. Baseline approaches

We introduce baseline models for the answer-selection task, which have been extensively studied and have proved their efficacy to the research community. These models are developed to compute the matching similarity between any pairs of text (the question and the target sentence in our case).

4.1.1. Compare Aggregate Framework (CompAggr).

This model (Wang and Jiang, 2016) computes the matching similarity between two texts (the question and the target sentence). It consists of attention, comparison, and aggregation parts.

Attention: The soft alignment of the question $\mathbf{Q} \in \mathbb{R}^{d \times Q}$ and target sentence $\mathbf{S} \in \mathbb{R}^{d \times S}$ (where d is a dimensionality of word embedding and Q and S are the length of the sequences in the question and sentence, respectively) is computed by applying an attention mechanism over the column vector in $\mathbf{Q}$ for each column vector in $\mathbf{S}$. With the computed alignment, we obtain a corresponding vector $\mathbf{A}^Q \in \mathbb{R}^{d \times S}$ as follows:

$$\mathbf{A}^Q = \mathbf{Q} \cdot \text{softmax}((\mathbf{WQ})^\intercal \mathbf{S}), \tag{1}$$

where $\mathbf{W}$ is a learned model parameter matrix.

Comparison: An element-wise multiplication is employed as a comparison function to combine each pair of $\mathbf{A}^Q$ and $\mathbf{S}$ into a vector $\mathbf{C} \in \mathbb{R}^{d \times S}$.

Aggregation: Kim (2014)'s CNN with n-types of filters is applied to aggregate all information in the vector $\mathbf{C}$. Finally, the model employs a fully connected layer to compute the matching score between the question and the target sentence as follows:

$$\begin{aligned} \mathbf{R} &= \text{CNN}(\mathbf{C}), \quad (\mathbf{R} \in \mathbb{R}^{nd}), \\ \hat{y}_c &= \text{softmax}((\mathbf{R})^\intercal \, \mathbf{W} + \mathbf{b}\,), \end{aligned} \tag{2}$$

where $\hat{y}_c$ is the predicted probability for the target class, c, and $\mathbf{W} \in \mathbb{R}^{nd \times c}$ and bias $\mathbf{b}$ are learned model parameters.

The loss function for the model is cross-entropy between predicted labels and true-labels as follows:

$$\mathcal{L} = -\log \sum_{i=1}^{N} \sum_{c=1}^{C} y_{i,c} \log(\hat{y}_{i,c}), \tag{3}$$

where $y_{i,c}$ is the true label vector and $\hat{y}_{i,c}$ is the predicted probability from the softmax layer. C is the total number of classes (true and false for this task), and N is the total number of samples used in training.

4.1.2. CompAggr-kMax.

This model (Bian et al., 2017) is an extension of the **CompAggr** model. The only differences lie in applying attention (in equation (1)) to both the $\mathbf{Q}$ and $\mathbf{S}$ side and applying k-max pooling before the softmax function as follows:

$$\begin{aligned} \mathbf{A}^Q &= \mathbf{Q} \cdot \text{softmax}(\text{kMax}((\mathbf{WQ})^\intercal \mathbf{S})), \\ \mathbf{A}^S &= \mathbf{S} \cdot \text{softmax}(\text{kMax}((\mathbf{WS})^\intercal \mathbf{Q})). \end{aligned} \tag{4}$$

4.1.3. CompClip-LM-LC.

This model (Yoon et al., 2019) is an extension of the **CompAggr-kMax** model. It employs the ELMo (Peters et al., 2018) model to enhance the word embedding layer for the question and target sentence by adopting the pretrained contextual language model. Additionally, it develops a latent clustering method to compute topic information in texts automatically and to use it as auxiliary information to improve the model performance.

4.1.4. IWAN.

This model (Shen et al., 2017a) is a variation model based on the compare aggregate framework. Unlike **CompAggr**, it employs RNNs to encode a sequence of the words in the text (question and target sentence independently). At the same time, it computes an inter-alignment weight between the question and the target sentence. The matching score is computed by aggregating this information (question, target sentence, and inter-aligned representation).

4.1.5. sCARNN.

This model (Tran et al., 2018) is an extension of the **IWAN** model. It proposes a novel recurrent unit to regulate the flow of the input (sequence of words in a text) and then replaces the RNNs in the **IWAN** model of the proposed unit.

4.2. Propagate-Selector

To build a model that can perform reasoning across multiple sentences, we propose a graph neural network model called **Propagate-selector** (**PS**). **PS** consists of four parts as follows (topology, node representation, aggregation, and update):

Topology: The topology of the graph determines the connections among the nodes in the graph. These connections will be used as a path that allows the information to flow from one node to another. Figure 2 depicts the topology of the proposed model. In an offline step, we organize the content of each instance in a graph, where each node represents a sentence from the passages and the question. Then, we add edges between nodes using the following topology:

- we fully connect nodes that represent sentences from the same passage (dotted-black);
- we fully connect nodes that represent the first sentence of each passage (dotted-red);
- we add an edge between the question and every node for each passage (dotted-blue).

In this way, we enable a path by which sentence nodes can propagate information between both inner and outer passages. Furthermore, we investigate different strategies for connecting those nodes in the graph and examine their corresponding effectiveness for detecting *supporting sentences* in the text (see the discussion in section 5.4.).

Node representation: Question $\mathbf{Q} \in \mathbb{R}^{d \times Q}$ and sentence $\mathbf{S}_i \in \mathbb{R}^{d \times S_i}$ (where d is the dimensionality of the word embedding and Q and S_i represent the lengths of the sequences in $\mathbf{Q}$ and $\mathbf{S}_i$, respectively) are processed to acquire the sentence-level information. Recent studies have shown that a pretrained language model helps the model capture

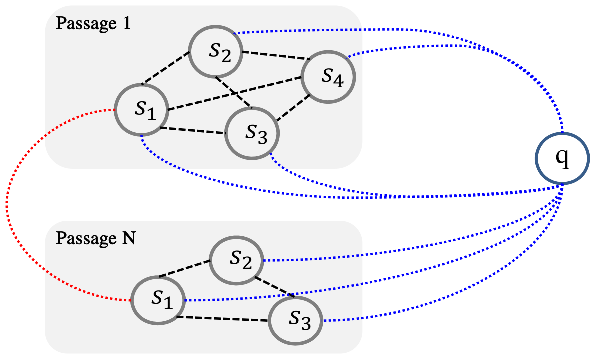


Figure 2: Topology of the proposed model. Each node represents a sentence from the passage and the question.

the contextual meaning of words in the sentence (Peters et al., 2018; Devlin et al., 2019). Following this study, we select an ELMo (Peters et al., 2018) language model for the word-embedding layer of our model as follows:

$$\begin{aligned} \mathbf{L}^Q &= \text{ELMo}(\mathbf{Q}),\ (\mathbf{L}^Q \in \mathbb{R}^{d \times Q}), \\ \mathbf{L}^S &= \text{ELMo}(\mathbf{S}),\ (\mathbf{L}^S \in \mathbb{R}^{d \times S}). \end{aligned} \tag{5}$$

Using these new representations, we compute the sentence representation as follows:

$$\begin{aligned} \mathbf{h}_t^Q &= f_\theta(\mathbf{h}_{t-1}^Q, \mathbf{L}_t^Q), \\ \mathbf{h}_t^S &= f_\theta(\mathbf{h}_{t-1}^S, \mathbf{L}_t^S), \\ \mathbf{N}^Q &= \mathbf{h}_{\text{last}}^Q,\ \ \mathbf{N}^S = \mathbf{h}_{\text{last}}^S, \end{aligned} \tag{6}$$

where f_θ is the RNN function with the weight parameters θ and $\mathbf{N}^Q \in \mathbb{R}^{d'}$ and $\mathbf{N}^S \in \mathbb{R}^{d'}$ are node representations for the question and sentence, respectively (where d' is the dimensionality of the RNN hidden units).

As computing the node representation is an essential process for acquiring information from a text, we investigate various approaches for encoding sentences, such as replacing the ELMo word representations using different methods (the GloVe (Pennington et al., 2014) or the BERT (Devlin et al., 2019)) and replacing the RNN function in equation (6) with the pooling method. Furthermore, we adopt the universal sentence encoding method based on the recently developed transformer model (Cer et al., 2018). Detailed information will be given in the section 5.5.

Aggregation: An iterative attentive aggregation function to the neighbor nodes is utilized to compute the amount of information to be propagated to each node in the graph as follows:

$$\begin{aligned} \mathbf{A}_v^{(k)} &= \sigma\Big(\sum_{u \in N(v)} a_{vu}^{(k)}\, \mathbf{W}^{(k)} \cdot \mathbf{N}_u^{(k)}\Big), \\ a_{vu}^{(k)} &= \frac{\exp(\mathbf{S}_{vu})}{\sum_k \exp(\mathbf{S}_{vk})}, \\ \mathbf{S}_{vu}^{(k)} &= (\mathbf{N}_v^{(k)})^\intercal \cdot \mathbf{W}^{(k)} \cdot \mathbf{N}_u^{(k)}, \end{aligned} \tag{7}$$

where $\mathbf{A}_v \in \mathbb{R}^{d'}$ is the aggregated information for the v-th node computed by attentive weighted summation of its neighbor nodes, a_{vu} is the attention weight between node *v* and its neighbor nodes u $(u \in N(v))$, $\mathbf{N}_u \in \mathbb{R}^{d'}$ is the *u*-th node representation, σ is a nonlinear activation function, and $\mathbf{W} \in \mathbb{R}^{d' \times d'}$ is the learned model parameter. Because all the nodes belong to a graph structure in which the iterative aggregation is performed among nodes, the *k* in the equation indicates that the computation occurs in the *k*-th hop (iteration).

Update: The aggregated information for the *v*-th node, $\mathbf{A}_v$ in equation (7), is combined with its previous node representation to update the node. We apply a skip connection to allow the model to learn the amount of information to be updated in each hop as follows:

$$\mathbf{N}_v^{(k)} = \sigma(\mathbf{W}' \cdot \{\mathbf{N}_v^{(k-1)}; \mathbf{A}_v^{(k)}\}), \qquad (8)$$

where σ is a nonlinear activation function, $\{;\}$ indicates vector concatenation, and $\mathbf{W}' \in \mathbb{R}^{d' \times 2d'}$ is the learned model parameter.

4.3. Optimization

Because our objective is to classify *supporting sentences* $(S_i \in P_n)$ from the given tuples $<Q, P_n, Y_i>$, we define two types of loss to be minimized. One is a rank loss that computes the cross-entropy loss between a question and each sentence using the ground-truth Y_i as follows:

$$\begin{aligned} \text{loss}_{rank} &= -\log \sum_{i=1}^{N} Y_i \log(S_i), \\ \text{S} &= [\text{score}_1, ..., \text{score}_i], \\ \text{score}_i &= g_\theta(\mathbf{N}^Q, \mathbf{N}_i^S), \end{aligned} \qquad (9)$$

where g_θ is a feedforward network that computes a similarity score between the final representation of the question and each sentence. The other is attention loss which is defined in each hop, to reward the model when it correctly attends the supporting sentences, as follows:

$$\text{loss}_{attn} = -\log \sum_{i=1}^{k} \sum_{i=1}^{N} Y_i \log(a_{qi}^{(k)}), \qquad (10)$$

where $a_{qi}^{(k)}$ indicates the relevance between the question node *q* and the *i*-th sentence node in the *k*-th hop as computed by equation (7).

Finally, these two losses are combined to construct the final objective function:

$$\mathcal{L} = \alpha \, \text{loss}_{rank} + \text{loss}_{attn}, \qquad (11)$$

where α is a hyperparameter.

5. Experiments

We regard the task as the problem of selecting the *supporting sentences* from the passages to answer the questions. Similar to the answer-selection task in the QA literature, we report the model performance using the mean average precision (MAP) and mean reciprocal rank (MRR) metrics. To evaluate the model performance, we use the HotpotQA dataset, which is described in section 3. Table 1 shows the properties of the dataset. We conduct a series of experiments to compare baseline methods with the newly proposed models. All source code developed to obtain the empirical results will be made available via a public web repository along with the dataset[1].

properties	**train**	**dev**
# questions	90,447	7,405
# sentences	3,703,344	306,487
passages / question	9.95	9.95
sentences / passage	4.12	4.16
sentences / question	40.94	41.39
supporting sentences / question	2.39	2.43
avg tokens (question)	17.92	15.83
avg tokens (sentence)	22.38	22.41

Table 1: Properties of the dataset

5.1. Implementation Details

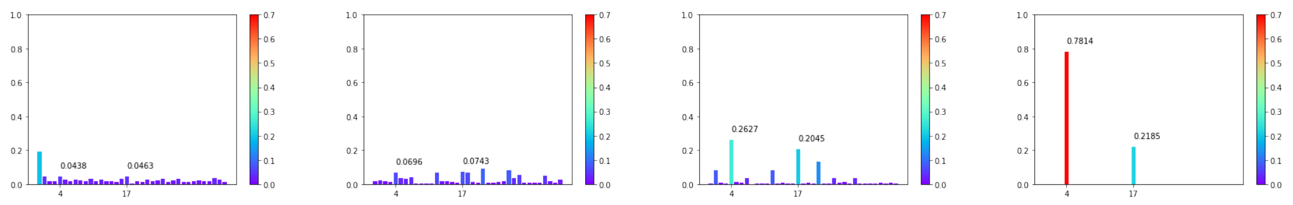
To implement the **propagate-selector** (**PS**) model, we first use a small version of ELMo (13.6 *M* parameters) that provides 256-dimensional context embedding. This choice was based on the available batch size (50 for our experiments) when training the complete model on a single GPU (GTX 1080 Ti). Then, we further experiment with the original version of ELMo (93.6 *M* parameters, 1024-dimensional context embedding). In this case, we were able to increase the batch size only up to 20, which results in excessive training time (approximately 90 hours). For the sentence encoding, we used a GRU with a hidden unit dimension of 200. The hidden unit weight matrix of the GRU is initialized using orthogonal weights (Saxe et al., 2013). Dropout (Srivastava et al., 2014) is applied for regularization purposes at a ratio of 0.7 for the GRU (in equation (6)) to 0.7 for the attention weight matrix (in equation (7)). For the nonlinear activation function (in equation (7) and (8)), we use the $tanh$ function.

Regarding the vocabulary, we replaced vocabulary with fewer than 12 instances in terms of term-frequency with "*UNK*" tokens. The final vocabulary size was 138,156. We also applied Adam optimizer (Kingma and Ba, 2014), including gradient clipping by norm at a threshold of 5.

5.2. Comparisons with Other Methods

Table 5.3. shows the model performances on the HotpotQA dataset. Because the dataset only provides training (trainset) and validation (devset) subsets, we report the model performances on these datasets. While training the model, we implement early termination based on the devset performance and measure the best performance. To compare the model performances, we choose widely used answer-selection models such as **IWAN** (Shen et al., 2017a), **sCARNN** (Tran et al., 2018), **CompAggr** (Wang and Jiang, 2016), **CompAggr-kMax** (Bian et al., 2017), and **CompClip-LM-LC** (Yoon et al., 2019), which were primarily developed to rank candidate answers for a given question (refer to the section 4.1. for detailed information on the models). In addition to the main proposed model, **PS**-*rnn-elmo*, we also report the model performance with a

[1]http://github.com/david-yoon/propagate-selector



(a) hop-1 (b) hop-2 (c) hop-3 (d) hop-4

Figure 3: Attention weights between the question and sentences in the passages. As the number of hops increases, the proposed model correctly classifies *supporting sentences* (ground-truth index 4 and 17).

Model	dev		train	
	MAP	MRR	MAP	MRR
IWAN [1]	0.526	0.680	0.605	0.775
sCARNN [2]	0.534	0.698	0.620	0.792
CompAggr [3]	0.659	0.812	0.796	0.911
CompAggr-kMax [4]	0.670	0.825	0.767	0.901
CompClip-LM-LC [5]	0.702	0.848	0.757	0.884
PS-*rnn-elmo-s*	0.716	0.841	0.813	0.916
PS-*rnn-elmo*	**0.734**	**0.853**	**0.863**	**0.945**

Table 2: Model performance on the HotpotQA dataset (top scores marked in bold). Models [1-5] are from (Shen et al., 2017a; Tran et al., 2018; Wang and Jiang, 2016; Bian et al., 2017; Yoon et al., 2019), respectively.

small version of ELMo, **PS**-*rnn-elmo-s*.
As shown in Table 5.3., the proposed **PS**-*rnn-elmo* shows a significant MAP performance improvement compared to the previous best model, **CompClip-LM-LC** (0.702 to 0.734 absolute).

5.3. Hop Analysis

Table 3 shows the model performance (**PS**-*rnn-elmo*) as the number of hops increases. We find that the model achieves the best performance in the 4-hop case but starts to degrade when the number of hops exceeds 4. We assume that the model experiences the vanishing gradient problem under a larger number of iterative propagations (hops). Table 4 shows the model performance with the small version of ELMo.
Figure 3 depicts the attention weight between the question node and each sentence node (hop-4 model case). As the hop number increases, we observe that the model properly identifies *supporting sentences* (in this example, sentences #4 and #17). This behavior demonstrates that our proposed model correctly learns how to propagate the necessary information among the sentence nodes via the iterative process.

5.4. Impact of Various Graph Topologies

The topology of the graph determines the path by which information flows and is aggregated. To see the quantitative contributions of each connection in the graph, we perform ablation experiments as follows:

- **Type-1**: We reduce the connections between sentences within the same passage. Only the previous and next sentences are connected to their neighbor sentence (see figure 4(a)).

- **Type-2**: We remove the connections between the passages, which reveals the contribution of the information flow among independent passages (see figure 4(b)).

- **Type-3**: We remove the connections between each sentence node and the question node (see figure 4(c)).

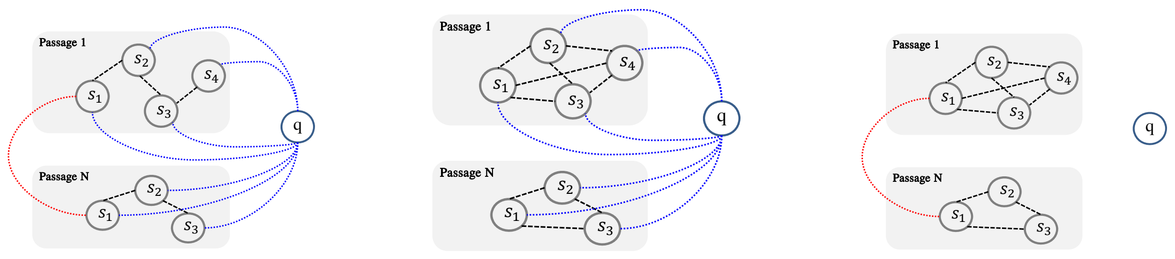
Figure 4 illustrates different types of connection strategies, and Table 5 shows their corresponding performances. To reach the best performance, we conduct experiments multiple times by changing the number of hops in the model from 1 to 6 for each case (Type-1 to Type-3). From the experiment, hop-4 is selected as the best-performing hyperparameter. However, all the model variations undergo performance degradation compared to the original topology (**PS**-*rnn-elmo-s*).

# hop	dev		train	
	MAP	MRR	MAP	MRR
1	0.651	0.794	0.716	0.842
2	0.653	0.797	0.721	0.850
3	0.698	0.830	0.800	0.908
4	**0.734**	**0.853**	**0.863**	**0.945**
5	0.700	0.827	0.803	0.906
6	0.457	0.606	0.467	0.621

Table 3: Model performance with original (5.5B) version of ELMo (top scores marked in bold) as the number of hop increases.

# hop	dev		train	
	MAP	MRR	MAP	MRR
1	0.648	0.790	0.708	0.842
2	0.655	0.801	0.720	0.853
3	0.681	0.816	0.768	0.886
4	0.706	0.834	0.796	0.906
5	**0.716**	**0.841**	**0.813**	**0.916**
6	0.441	0.596	0.452	0.600

Table 4: Model performance with small version of ELMo (top scores marked in bold) as the number of hop increases.



(a) Type-1 (b) Type-2 (c) Type-3

Figure 4: Different typologies for the graph. Type-1 reduce connection within the passage, type-2 remove connection between the passages and type-3 remove connection between each sentence node and the question node.

Model	dev		train	
	MAP	MRR	MAP	MRR
PS-*rnn-elmo-s*	**0.716**	**0.841**	**0.813**	**0.916**
Type-1 (*rnn-elmo-s*)	0.694	0.834	0.807	0.915
Type-2 (*rnn-elmo-s*)	0.705	0.836	0.792	0.903
Type-3 (*rnn-elmo-s*)	0.658	0.796	0.729	0.857

Table 5: Model performance with different typologies. The connection strategies between nodes for each type are illustrated in figure 4.

5.5. Impact of Node Representation

To see the effectiveness of the various approaches for computing sentence representation, we investigate combinations of well-studied methods.

5.5.1. Word Representation

Vector representations of the words in each sentence are computed from the original version of the ELMo model (-elmo), a small version of the ELMo model (-elmo-s), BERT (Devlin et al., 2019) model (-bert), or mapped to GloVe word embedding (-glove).

5.5.2. Node Representation

Each node representation is computed by employing three general methods for encoding the sequence of word representations as follows:

- We employ an RNN model (-rnn) to encode sequential information in the sentence. The final representation of the RNN's hidden status is considered as a node representation (see equation (6)).

- We apply a pooling method (-avg) that averages all the word representations in the sentence to compute the node representation as follows: $\mathbf{N}^Q = \text{average}(\mathbf{Q})$, $\mathbf{N}^S = \text{average}(\mathbf{S})$. These new representations-$\mathbf{N}^Q$ and $\mathbf{N}^S$-are substituted for the node representations in equation (6).

- We adopt the pretrained universal sentence-encoding method (-USD_T), which is based on the recently developed transformer model (Cer et al., 2018). This model computes sentence representation directly from the sequence of words in any text.

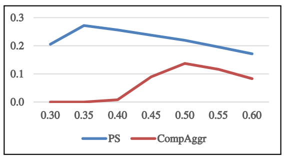
Model	dev		train	
	MAP	MRR	MAP	MRR
PS-*USD_T*	0.651	0.795	0.693	0.830
PS-*avg-glove*	0.617	0.753	0.876	0.945
PS-*avg-elmo-s*	0.471	0.611	0.483	0.625
PS-*rnn-glove*	0.700	0.822	**0.919**	**0.971**
PS-*rnn-elmo-s*	0.716	0.841	0.813	0.916
PS-*rnn-elmo*	**0.734**	**0.853**	0.863	0.945
PS-*rnn-bert*	0.667	0.806	0.708	0.841

Table 6: Model performance with the different method for computing node representation.

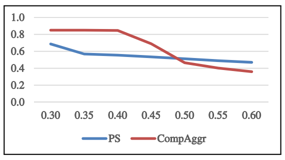
Table 6 depicts the model performance with different node representation methods. In all cases, the RNN encoding skims (-rnn) performs better than that of the average pooling (-avg). Interestingly, average pooling with ELMo representation (**PS**-*avg-elmo-s*) performs worse than in the GloVe representation (**PS**-*avg-glove*) case. From this result, we find that averaging ELMo does not produce proper node representations. For the **PS**-*rnn-bert* case, we do not fine-tune the BERT model and only use its computing word representation. We expect there exists a possibility to enhance model performance by fine-tuning the BERT with the end-to-end training process.

6. Discussion

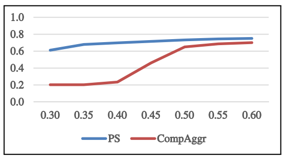
In this study, we focus on a model that can detect *supporting sentences* to answer a question. We do not consider competing against the full QA systems, i.e., machine reading QA (MRQA) models, which are jointly trained with two objectives, "*extracting answer span*" and "*detecting supporting sentence*." Note that we do not use the exact "answer span" information when detecting the *supporting sentences*. We think "answer-span" supervision allows the model to track the *supporting sentences* from simple word matching. Therefore, our investigations are focused on evaluating and analyzing the effectiveness of the proposed graph neural network-based model for classifying *supporting sentences* compared to the well-known answer-selection QA models. To evaluate the performance of the proposed model from a different perspective, we adopt other traditional measures for the QA system (i.e.,



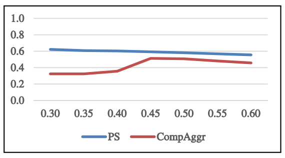
(a) exact match



(b) recall



(c) precision



(d) f1

Figure 5: Model performance with various measure. The x-axis shows a threshold value that is used for determining the label of the question-supporting sentence pair by the confidence score.

precision, recall, and f1), and evaluate our methods. These measures require a specific label (*true* or *false*) for each pair of data (the question and the supporting sentence candidate). As our model computes the confidence scores for each pair of data, we give a *true*-label when the confidence scores are greater than a predefined threshold value (otherwise, we give the pair a *false*-label). Figure 5 shows the model performances (**PS**-*rnn-elmo* vs **CompAggr**) in regards to the variation of the threshold value (0.3 to 0.6).
In future research directions, we will investigate the best way to combine our proposed model with existing MRQA algorithms to build a full QA system. It would also be possible to link the current graph to another graph (i.e., knowledge graph) to engage external knowledge information in the question-answering system. We also hope that our work inspires future works aiming to perform multihop reasoning on the free-form text.

7. Conclusion

In this paper, we propose a graph neural network that finds the sentences crucial for answering a question. The experiments demonstrate that the model correctly classifies *supporting sentences* by iteratively propagating the necessary information through its novel architecture. We believe that our approach will play an important role in building a QA pipeline in combination with other MRQA models trained in an end-to-end manner.

8. Acknowledgements

K. Jung is with ASRI, Seoul National University, Korea. This work was supported by the Ministry of Trade, Industry & Energy (MOTIE, Korea) under Industrial Technology Innovation Program (No.10073144) and by the NRF grant funded by the Korea government (MSIT) (NRF2016M3C4A7952587).

9. Bibliographical References

Bian, W., Li, S., Yang, Z., Chen, G., and Lin, Z. (2017). A compare-aggregate model with dynamic-clip attention for answer selection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1987–1990. ACM.

Cer, D., Yang, Y., Kong, S.-y., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., et al. (2018). Universal sentence encoder. *arXiv preprint arXiv:1803.11175.*

Clark, C. and Gardner, M. (2018). Simple and effective multi-paragraph reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 845–855.

De Cao, N., Aziz, W., and Titov, I. (2019). Question answering by reasoning across documents with graph convolutional networks. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2306–2317.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186.

Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., and Yin, D. (2019). Graph neural networks for social recommendation. In *The World Wide Web Conference*, pages 417–426. ACM.

Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034.

Joshi, M., Choi, E., Weld, D., and Zettlemoyer, L. (2017). Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1601–1611.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR).*

Palm, R., Paquet, U., and Winther, O. (2018). Recurrent relational networks. In *Advances in Neural Information Processing Systems*, pages 3368–3378.

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

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2227–2237.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Saxe, A. M., McClelland, J. L., and Ganguli, S. (2013). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*.

Seo, M., Kembhavi, A., Farhadi, A., and Hajishirzi, H. (2016). Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations (ICLR)*.

Shen, G., Yang, Y., and Deng, Z.-H. (2017a). Inter-weighted alignment network for sentence pair modeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1179–1189.

Shen, Y., Huang, P.-S., Gao, J., and Chen, W. (2017b). Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1047–1055. ACM.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958.

Tran, Q. H., Lai, T., Haffari, G., Zukerman, I., Bui, T., and Bui, H. (2018). The context-dependent additive recurrent neural net. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 1274–1283.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. (2018). Graph attention networks. In *International Conference on Learning Representations (ICLR)*.

Wang, S. and Jiang, J. (2016). A compare-aggregate model for matching text sequences. *arXiv preprint arXiv:1611.01747*.

Wang, W., Yang, N., Wei, F., Chang, B., and Zhou, M. (2017). Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 189–198.

Wang, S., Yu, M., Guo, X., Wang, Z., Klinger, T., Zhang, W., Chang, S., Tesauro, G., Zhou, B., and Jiang, J. (2018). R 3: Reinforced ranker-reader for open-domain question answering. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

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

Welbl, J., Stenetorp, P., and Riedel, S. (2018). Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association of Computational Linguistics*, 6:287–302.

Wu, Y., Liu, X., Feng, Y., Wang, Z., Yan, R., and Zhao, D. (2019). Relation-aware entity alignment for heterogeneous knowledge graphs. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 5278–5284. AAAI Press.

Xiong, C., Zhong, V., and Socher, R. (2016). Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*.

Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W., Salakhutdinov, R., and Manning, C. D. (2018). HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.

Yoon, S., Dernoncourt, F., Kim, D. S., Bui, T., and Jung, K. (2019). A compare-aggregate model with latent clustering for answer selection. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2093–2096.

Zitnik, M., Agrawal, M., and Leskovec, J. (2018). Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466.