Machine Intelligence Research
www.mi-research.net

21(2), April 2024, 272-282
DOI: 10.1007/s11633-023-1421-0

GraphFlow+: Exploiting Conversation Flow in Conversational Machine Comprehension with Graph Neural Networks

Jing Hu[1*] Lingfei Wu[2*] Yu Chen[3] Po Hu[1] Mohammed J. Zaki[4]

[1]School of Computer Science, Central China Normal University, Wuhan 430079, China
[2]Pinterest, San Francisco 94016, USA
[3]Meta, Mountain View 94039, USA
[4]Rensselaer Polytechnic Institute, Troy 12180, USA

Abstract: The conversation machine comprehension (MC) task aims to answer questions in the multi-turn conversation for a single passage. However, recent approaches don't exploit information from historical conversations effectively, which results in some references and ellipsis in the current question cannot be recognized. In addition, these methods do not consider the rich semantic relationships between words when reasoning about the passage text. In this paper, we propose a novel model GraphFlow+, which constructs a context graph for each conversation turn and uses a unique recurrent graph neural network (GNN) to model the temporal dependencies between the context graphs of each turn. Specifically, we exploit three different ways to construct text graphs, including the dynamic graph, static graph, and hybrid graph that combines the two. Our experiments on CoQA, QuAC and DoQA show that the GraphFlow+ model can outperform the state-of-the-art approaches.

Keywords: Conversational machine comprehension (MC), reading comprehension, question answering, graph neural networks (GNNs), natural language processing (NLP).

Citation: J. Hu, L. Wu, Y. Chen, P. Hu, M. J. Zaki. GraphFlow+: Exploiting conversation flow in conversational machine comprehension with graph neural networks. *Machine Intelligence Research*, vol.21, no.2, pp.272–282, 2024. http://doi.org/10.1007/s11633-023-1421-0

1 Introduction

In the field of machine comprehension (MC), conversation machine comprehension has recently received much attention[1]. Unlike most of the traditional research in MC which revolves around answering a single question for a given text, conversational MC expects the machine to understand a passage and answer several questions in multi-turn conversation. Although there are many great studies(e.g., SQuAD[2]) in single-turn MC, conversational MC would be more promising because it is more in line with the way humans conduct conversations in their daily lives.

Therefore, there are many challenges specific to multi-turn conversational MC. Firstly, the focus of the text always migrates as the conversation proceeds[3, 4]. Secondly, caused by references and ellipsis, the question in the current turn may need to refer back to the conversation history.

To solve these challenges, the previous approaches were divided into two ways. The first way was to add the conversation history to the front of the current question[4, 5], and the other way was to add the previous answer position to the passage[3, 6]. Both approaches regard the task as an MC task of a single turn without considering modelling the conversation flow. In addition, Huang et al.[7] introduce the concept of Flow and proposes the integration-Flow (IF) layers, which can be combined with intermediate representations generated by conversation histories.

Nevertheless, the IF mechanism has several limitations during the reasoning process. Firstly, since the IF mechanism does not combine the results of previous reasoning processes into the current one immediately, it is not so valid in terms of interweaving the two processing directions of passage words and question turns. Instead, all reasoning processes are carried out in parallel. Thus,

Research Article
Special Issue on Commonsense Knowledge and Reasoning: Representation, Acquisition and Applications
Manuscript received on December 6, 2022; accepted on February 2, 2023; published online on January 12, 2024
Recommended by Associate Editor Jun Zhao
Colored figures are available in the online version at https://link.springer.com/journal/11633
* These authors contribute equally to this work
© Institute of Automation, Chinese Academy of Sciences and Springer-Verlag GmbH Germany, part of Springer Nature 2024

the results of previous reasoning processes cannot improve the reasoning performance on the current turn. Secondly, they also do not make use of the rich semantic relationships between words but merely treat the passage as a sequence of words. Recently, some works on multi-hop MC[8, 9] have revealed that constructing a context graph and processing it using graph neural network (GNN) is better than processing a sequence of words through a recurrent neural network (RNN).

In this work, we proposed a novel model GraphFlow+, which enhances our previous work GraphFlow[10]. It is based on the context graph and GNN[11]. For graph construction, we use several different ways to construct the context graph, including the dynamic graph, static graph, and hybrid graph combined of both. In the dynamic graph, we regard each word as a graph node and propose an approach to construct the conversation history-aware context graph. For the construction of static graphs, we used the current mainstream static graphs, including the dependency graph and constituency graph. In addition, we explore the integration of static and dynamic graphs to exploit the explicit dependencies and semantic information contained in static graphs and the implicit information learned in dynamic graphs. We also propose a novel recurrent graph neural network (RGNN) to implement the flow mechanism, which can model the temporal dependencies between the context graphs of each turn.

In summary, our contributions to this paper are as follows:

1) We propose the GraphFlow+, a GNN and context graph-based model, which can capture conversational flow for a passage in conversational MC.

2) We exploit several novel ways to construct the graph, including dynamic conversation history aware context graph, a variety of commonly used static graphs, and the hybrid graph of the two. In addition, we apply a novel RGNN to implement the flow mechanism, which can model the temporal dependencies between the context graphs of each turn.

3) We conduct some experiments on CoQA, QuAC, and DoQA benchmarks, where our proposed method surpasses the state-of-the-art approaches. Additionally, we demonstrate the interpretability of our model's reasoning process through visualization experiments.

2 Approach

2.1 Problem formulation

The objective of conversational MC is to identify a span within the context that answers the question at each turn. The context, represented as a sequence of m words, is denoted as $C = \{c_1, c_2, \cdots, c_m\}$. The question at the i-th turn, consisting of n^i words, is represented as $Q^i = \{q_1^{(i)}, q_2^{(i)}, \cdots, q_{n^i}^{(i)}\}$. The number of turns in a conversation is T.

2.2 Model architecture

2.2.1 Overview

A graphical representation of the model overview can be seen in Fig. 1. We will describe in detail the four parts of the model: 1) Encoding layer, where the model encodes the question and context; 2) Context graph learning, where the model constructs the context graphs in three different ways; 3) Reasoning layer, where the model uses a novel RGNN to model the temporal dependencies between the context graphs of each turn; 4) Prediction layer, where the model computes the start and end probabilities for all tokens to predict the answer.

2.2.2 Encoding layer

The encoding layer encodes the context and question words in each turn. For the i-th turn, the embedding of context word c_j is represented as a vector $w_{c_j}^{(i)}$, which is a concatenation of linguistic vector $f_{\text{ling}}(c_j^{(i)})$, word embedding, aligned vector $f_{\text{aligned}}(c_j^{(i)})$ and answer vector $f_{\text{ans}}(c_j^{(i)})$. Similarly, the question word $q_k^{(i)}$ is encoded as a

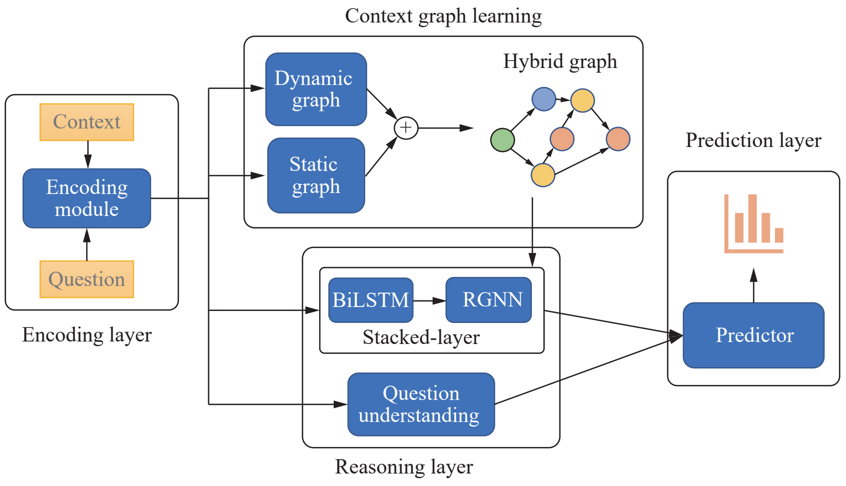


Fig. 1 GraphFlow+ model structure

vector $w_{q_k}^{(i)}$ that concatenates the word embedding and turn marker vector $f_{\text{turn}}(q_k^{(i)})$. Below is a specific description of each section.

Linguistic features

For the context word c_j, we use named entity recognition, part-of-speech and exact matching to represent the linguistic feature of context at turn i, where exact matching denotes whether c_j appears in $Q^{(i)}$. Then, we concatenate these features to a vector $f_{\text{ling}}(c_j^{(i)})$.

Pretrained word embeddings

For context word c_j and question word $q_k^{(i)}$, since large-scale pretrained word vectors can be rich in potential information, we embed all context and question using the pretrained GloVe embeddings[12] and BERT embeddings[13].

Aligned question embeddings

For context word c_j and the question word q_k, we learn a soft alignment between context and question by applying an attention mechanism. For the purpose of simplification, the turn index i is eliminated. Following [14], the aligned question embedding at each turn is calculated as follows:

$$f_{\text{aligned}}(c_j) = \sum_k \beta_{j,k} g_k^Q \tag{1}$$

where g_k^Q represents the GloVe embedding for word q_k. The attention score between c_j and q_k, represented as $\beta_{j,k}$, is calculated as follows:

$$\beta_{j,k} \propto \exp(\text{ReLU}(Wg_j^C)^{\text{T}}\text{ReLU}(Wg_k^Q)) \tag{2}$$

where W denotes a trainable matrix, d is the hidden state size, g_j^C and g_k^Q are the GloVe embeddings of c_j and q_k, respectively.

We simplify the above process as $\text{Align}(X, Y, Z)$, indicating that we combine the vector set Z after calculating the attention score between two vector sets X and Y. Thus, the above process is simplified as

$$f_{\text{aligned}}(C) = \text{Align}(g^C, g^Q, g^Q). \tag{3}$$

Conversation history

For each word c_j, we follow Choi's[3] approach to context word embedding using a vector $f_{\text{ans}}(c_j^{(i)})$ to encode previous N answer locations. In addition, we concatenate a turn marker embedding $f_{\text{turn}}(q_k^{(i)})$ to each word vector in the extended question, signalling the turn to which the word belongs.

For simplicity, we denote $W_C^{(i)}$ and $W_Q^{(i)}$ as a sequence of context word vectors $w_{c_j}^{(i)}$ and question word vectors $w_{q_k}^{(i)}$, respectively.

2.2.3 Context graph learning

To apply the GNNs to the conversational MC task, graph construction is an extremely significant step. Here, we introduce three approaches to constructing the context graph.

Dynamic context graph

We propose a way to construct a dynamic context graph that takes into account the conversation history at each turn. When considering the flow of conversation, we will find that the context graph may change across different turns. However, the existing approaches[8, 9, 15] using GNNs rely on either manually constructed or ground-truth graphs, which may lead to various problems. Firstly, manually generated graphs may have many labelling and connectivity errors, which can lead to poor results. Secondly, ground-truth graphs are not constantly available. Therefore, we construct our conversation history aware graph dynamically from the original context at each turn. In this graph, each token in context is a node, and the structure is in dynamic change.

To specify, we use an attention mechanism on the context representations $W_C^{(i)}$ to compute an attention matrix $A^{(i)}$, which can be employed as a weighted adjacency matrix of the context graph:

$$A_{\text{dynamic}}^{(i)} = (W_C^{(i)} \odot u)^{\text{T}} W_C^{(i)} \tag{4}$$

where $\odot$ is the element-wise multiplication, and u is a trainable weight vector with d_c dimensions that is able to identify significant dimensions in $w_{c_j}^{(i)}$, which has a d_c dimension.

However, the complete graph derived from the weighted adjacency matrix is too complex. It is not only computationally expensive but also retains some unimportant edges which may lead to poor results. Therefore, we simplify this complete graph by using a KNN-style approach to generate a sparse graph. For each point, we only keep the k most significant edges connected to it. Finally, we apply a softmax function on the elements of the KNN-style adjacency matrix to obtain a normalized adjacency matrix.

$$\widetilde{A}_d^{(i)} = \text{softmax}(\text{topk}(A_d^{(i)})). \tag{5}$$

It is noteworthy that because the K nearest attention score is retained to calculate the final normalized adjacency matrix, the supervised signal can still back-propagate through the sparsification module.

Static context graph

While the graph learner can learn a nice dynamic graph structure to capture the changes in the conversation flow, there are still some limitations in that the graph learner cannot capture the rich semantic relations between useful objects in the passage context, which is the foundation for performing graph-based complex reasoning. Therefore, we construct a static graph to obtain the information between nodes.

The dependency graph and constituency graph are two widely used static graphs[1], which can represent the relationship between each word in a sentence. A dependency graph captures the dependencies between different objects in a given sentence, and it adds edges between adjacent words in the context in order, in addition to adding edges between the dependencies. On the other hand, the constituency graph can capture phrase-based syntactic relations in a given sentence. There are three types of nodes in the constituency graph, the word token node, the subject node, and the part-of-speech (POS) node. Since the context is always long, we reduce the size of the constituency graph by removing nodes of the POS type to improve the efficiency of the reasoning process. This pruning method can keep the structure of the original graph unchanged to the greatest extent, and the rich semantic information in the original graph will not be lost.

Hybrid graph

To learn a better graph structure, we combine the dynamic graph with the static graph to obtain a hybrid graph structure, which has both rich semantic information and conversation history-aware information. In addition, the static graph can only capture the grammatical relationship at the sentence and phrase level, but not the relationship between different sentences of the whole article. The dynamic graph constructs the graph in the case of regarding every token as a node, which can capture the passage context information. Therefore, the hybrid graph can alleviate the disadvantages brought by using only static graphs. Inspired by Chen et al.[16] on combining two types of different graphs, we use a hyperparameter to obtain a hybrid weighted adjacency matrix, which is computed by

$$A_c^{(i)} = \lambda A_s + (1-\lambda)\widetilde{A}_d^{(i)} \tag{6}$$

where A_s is the adjacency matrix of the static graph, and $A_d^{(i)}$ is the dynamic graph learned at turn i. λ is a hyperparameter that can range from 0 to 1, which can balance the trade-off between the dynamic structure and the static graph structure at each turn. The output $A_c^{(i)}$ is used as the weighted adjacency matrix of the hybrid graph at turn i.

At each turn, we use this approach to combine the current dynamic context graph with the pre-generated static context graph.

2.2.4 Reasoning layer

After obtaining the graph structure, the next step is to reason through the conversation flow under the guidance of the graph and the embedded question and context.

Question understanding

At turn i, we encode the question $Q^{(i)}$ using question embedding $W_Q^{(i)}$ with bidirectional LSTM[17] to obtain contextualized embedding $Q^{(i)} \in \mathbf{R}^{d\times n}$.

$$\boldsymbol{Q}^{(i)} = q_1^{(i)}, \cdots, q_n^{(i)} = \mathrm{BiLSTM}(W_Q^{(i)}). \tag{7}$$

We then employ a self-attention mechanism on question embedding to obtain a weighted sum of each word vector for each question.

$$\widetilde{q}^{(i)} = \sum_k a_k^{(i)} q_k^{(i)}, \quad \text{where} \quad a_k^{(i)} \propto \exp(w^{\mathrm{T}} q_k^{(i)}) \tag{8}$$

where w is a trainable vector with a dimension of d.

Then, we employ an LSTM to encode the questions to generate history-aware question vectors.

$$p^{(1)}, \cdots, p^{(T)} = \mathrm{LSTM}(\widetilde{q}^{(1)}, \cdots, \widetilde{q}^{(T)}) \tag{9}$$

Then, the hidden states from LSTM network $p^{(1)}, \cdots,$ $p^{(T)}$ will be used for predicting answers in the prediction layer.

Graph reasoning

In the graph reasoning module, we combine the results of the previous reasoning process into the current reasoning process. Now we have a sequence containing the context graph at each turn in order.

To process a sequence of context graphs, we propose a novel RGNN which exploits the idea of RNN to deal with the sequence and exploits the idea of GNN to deal with every context graph. Fig. 2 shows its architecture. As we progress through a sequence of graphs, we employ a shared GNN cell on the context graph and pass the output of the previous GNN cell to the current context graph as part of the input, similar to an RNN. Our proposed RGNN module combines the strengths of RNNs and GNNs, allowing it to effectively perform sequential learning and relational reasoning.

Here is the precise procedure for RGNN. Initially, we employ a BiLSTM to encode the $W_c^{(i)}$ that was obtained in the previous encoding layer.

$$C^{(i)} = \mathrm{BiLSTM}(W_C^{(i)}). \tag{10}$$

At turn i, the vector $C^{(i)}$ is used as the initial node embedding. Then, we fuse the initial context node embedding $C^{(i)}$ with the updated node embedding $\bar{C}^{(i-1)}$ at the last turn and apply a parameter-sharing GNN to the context graph $\mathcal{G}^{(i)}$ to obtain the updated node embedding at this turn.

$$\bar{C}^{(i)} = \mathrm{GNN}(\mathrm{Fuse}(C^{(i)}, \bar{C}^{(i-1)}), \widetilde{A}^{(i)}). \tag{11}$$

At the first turn, we set $\bar{C}^{(0)} = C^0$ because there is no conversation history available. The fusion function, Fuse, is designed to combine two inputs using a gating mechanism.

$$\begin{aligned} &\mathrm{Fuse}(a,b) = z\times a + (1-z)\times b \\ &z = \sigma(W_z[a;b;a\times b;a-b] + b_z) \end{aligned} \tag{12}$$

where W_z and b_z are the learnable matrix and vector, z is

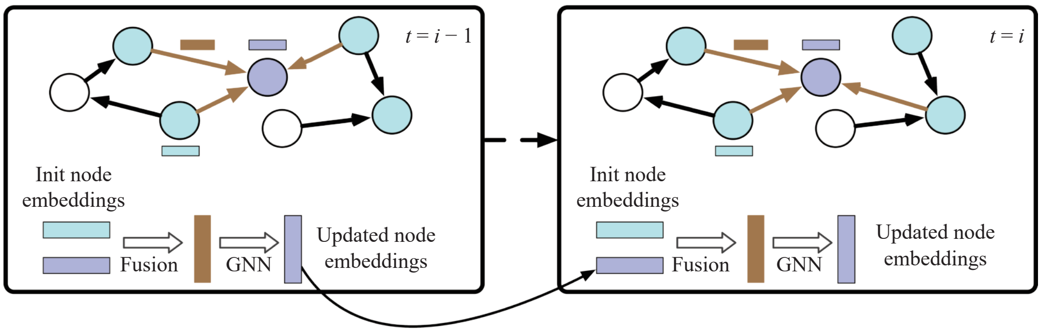


Fig. 2 Proposed RGNN module architecture

a gating vector, and σ is a sigmoid function.

Our framework is flexible and can be used with different types of GNN cells. In this paper, we use gated graph neural networks (GGNN)[18] to capture long-range dependencies in graphs using multi-hop message passing. GGNN uses the GRU[19] to update the node embeddings based on the weighted sum of its neighboring node embeddings. The final node embeddings are obtained from the last hop of message passing.

To make the notation easier, we refer to the RGNN module as

$$\bar{C}^{(i)} = \mathrm{RGNN}(C^{(i)}, \widetilde{A}^{(i)}) \tag{13}$$

where $i = 1, \cdots, T$. We input the node embeddings $\{C^{(i)}\}_{i=1}^{T}$ and the normalized adjacency matrices $\{\widetilde{A}^{(i)}\}_{i=1}^{T}$ at each turn into the module, and obtain the updated node embeddings $\{\bar{C}^{(i)}\}_{i=1}^{T}$ as output.

Following the recent work[20], we utilize the following module that applies stacked RGNN layers to low-level and high-level representations of the context separately, with two RGNN layers used for each representation.

$$\begin{aligned}
H_C^{(i)} &= [\bar{C}^{(i)}; g^C; \mathrm{BERT}^C] \\
H_Q^{(i)} &= [Q^{(i)}; g^{Q^{(i)}}; \mathrm{BERT}^{Q^{(i)}}] \\
f^2_{\mathrm{aligned}}(C^{(i)}) &= \mathrm{Align}(H_C^{(i)}, H_Q^{(i)}, Q^{(i)}] \\
\hat{C}^{(i)} &= \mathrm{BiLSTM}([\bar{C}^{(i)}; f^2_{\mathrm{aligned}}(C^{(i)})]) \\
\widetilde{C}^{(i)} &= \mathrm{RGNN}(\hat{C}^{(i)}, \widetilde{A}^{(i)}),\ i = 1, \cdots, T
\end{aligned} \tag{14}$$

where $\{\widetilde{C}^{(i)}\}_{i=1}^{T}$ is the final context representation.

2.2.5 Prediction layer

At turn i, the prediction layer predicts the span of the answer by calculating the probabilities of the j-th word being either the starting or ending word of the span. For simplicity, we omit the turn index i in this module. The start probability P_j^S is computed as follows:

$$P_j^S \propto \exp(\widetilde{c}_j^{\mathrm{T}} W_S p) \tag{15}$$

where $\widetilde{c}_j$ is the final context representation of the j-th word, p is the question representation, and W_S is a trainable matrix. Next, we pass p to a GRU cell by incorporating the context summary and converting it to $\widetilde{p}$.

$$\widetilde{p} = \mathrm{GRU}(p, \sum_j P_j^S \widetilde{c}_j). \tag{16}$$

Then, the end probability P_j^E is calculated by

$$P_j^E \propto \exp(\widetilde{c}_j^{\mathrm{T}} W_E \widetilde{p}) \tag{17}$$

where W_E is a $d \times d$ trainable weight.

Since the answers to some of the questions in the dataset we use as a benchmark are free-form text, rather than a span, we follow previous work and adopt an extractive approach to handle non-extractive questions.

Specifically, among all the questions that cannot be answered with the span, there are some answers that consist only of yes or no, and some answers that are completely free-form. We apply a classifier to determine the answer type, including whether the answer can be answered and whether the answer is a span in the original text. Then we convert the free-form answer into predefined text in the dataset. If the classifier determines that the answer is in a free-form format, we will utilize the pre-defined text as the answer. The probability of the answer type is calculated by

$$P^C = \sigma(f_c(p)[f_{\mathrm{mean}}(\widetilde{C}); f_{\max}(\widetilde{C})]^{\mathrm{T}}) \tag{18}$$

where f_c is a dense layer that converts a d-dimensional vector to a (num_class $\times\, 2d$)-dimensional vector. The σ function consists of both a sigmoid and a softmax function. $f_{\mathrm{mean}}(\cdot)$ and $f_{\max}(\cdot)$ represent the average pooling and max pooling functions, respectively.

2.3 Training and inference

At turn i, we train our model with the cross entropy loss of both answer type prediction and text span prediction. For simplicity, we omit the turn index i,

$$\mathcal{L} = -I^S(\log(P_s^S) + \log(P_e^E)) + \log P_t^C \tag{19}$$

where I^S represents whether the answer to the question is a span from the text. The ground truth positions for the start and end of the answer span are represented by s and e, respectively, and the answer type is represented by t.

When making predictions, we use P^C to determine if the answer to the question is a span from the text. If the prediction is affirmative, we select the span with maximum $P^S_{\hat{s}}$, $P^E_{\hat{e}}$ subject to a maximum span length threshold and predict the span as $\hat{s}$, $\hat{e}$.

3 Experiments

3.1 Dataset, baseline and metric

Our experiments utilize three benchmark datasets: DoQA[21], CoQA[4], and QuAC[3].

The CoQA dataset comprises 127 000 questions and answers from 8 000 conversations, with the answer being free-form and not limited to a contextual text span. Approximately 33.2% of the questions are answered in an abstract manner. The average question length is 5.5 words, and the average number of turns per conversation is 15.2 words. The QuAC dataset comprises 98 000 questions and answers from 13 000 conversations, with all answers being a span of context. The average question length is 6.5 words, with an average of 7.2 questions per dialogue. The DoQA dataset is a collection of questions and answers from conversations within the cooking domain. It includes 7 300 questions and answers from 1 600 conversations. Similar to CoQA, not all answers are spans of the original text, and 31.3% of the answers are free-form.

Then we compare our model with some baselines: PGNet[22], DrQA[23], DrQA+PGNet[4], BiDAF++[6], FlowQA[7], SDNet[5], BERT[13] and Flow (unpublished).

For evaluation metrics, we adopt the F1 score for all benchmarks and human equivalence score (HEQ) for QuAC and DoQA. The F1 score measures the balance between precision and recall. HEQ can compare our model's performance with that of an average human. The answer from the model is regarded as correct when its F1 score exceeds the average human F1 score. And HEQ-Q and HEQ-D calculate the accuracy for each question and dialogue respectively. For CoQA, the exact match (EM) is also used as a metric.

3.2 Model settings

We will introduce the implementation details in our model. The dimensions for the embeddings of POS, NER, exact matching, and turn marker are 12, 8, 3 and 3, respectively. And the BERT embedding is calculated by the sum of the BERT layer outputs, with a size of 300. For graph construction, the sparsification module uses a value of 15 for K and the hybrid graph module uses a value of 0.17 for λ. In addition, we use CoreNLP[24] to build the static graph. The GNN module has a hop number of 5 for CoQA and DoQA, and 3 for QuAC. For the hyperparameters over the model, which we tuned on the development set, dropout is set to 0.3 for RNN layers and GloVe, 0.4 for BERT. All experiments are optimized by Adamax[25] with a learning rate of 0.001.

3.3 Experimental results

The performance of our model with a dynamic graph and baseline models is displayed in Tables 1–3, demonstrating that our model surpasses the existing state-of-the-art baselines. Our model outperforms FlowQA on CoQA, QuAC, and DoQA, with a 2.3%, 0.8%, and 2.5% increase in F1, respectively. This indicates that our RGNN-based flow mechanism is superior to the IF mechanism. In addition, our model also outperforms ZDNet on CoQA with a 0.7% increase in F1, even though ZDNet uses inter-attention and self-attention mechanisms.

Table 3 shows the results of the influence of different graph types. We test the dynamic graph, static graph (i.e., dependency graph, constituency graph), and hybrid graph combined with dynamic graph and dependency graph. From Table 3, we can see that on the DoQA dataset, the hybrid graph improves F1 by 1.8% compared to the dependency graph and 0.8% compared to the dynamic

Table 1 F1 results (%) of models and humans on the CoQA test set

	Children's story	Literature	Mid/High school exams	News	Wiki	Reddit	Science	Overall
PGNet	49.0	43.3	47.5	47.5	45.1	38.6	38.1	44.1
DrQA	46.7	53.9	54.1	57.8	59.4	45.0	51.0	52.6
DrQA+PGNet	64.2	63.7	67.1	68.3	71.4	57.8	63.1	65.1
BiDAF++	66.5	65.7	70.2	71.6	72.6	60.8	67.1	67.8
FlowQA	73.7	71.6	76.8	79.0	80.2	67.8	76.1	75.0
Flow [Unpublished]	–	–	–	–	–	–	–	75.8
SDNet	75.4	73.9	77.1	**80.3**	**83.1**	69.8	76.8	76.6
GraphFlow (dynamic)	**77.1**	**75.6**	**77.5**	79.1	82.5	**70.8**	**78.4**	**77.3**
Human	90.2	88.4	89.8	88.6	89.9	86.7	88.1	88.8

Table 2 F1, HEQ-Q, and HEQ-D results (%) of models and humans on the QuAC test set

	F1	HEQ-Q	HEQ-D
BiDAF++	60.1	54.8	4.0
FlowQA	64.1	59.6	**5.8**
GraphFlow (dynamic)	**64.9**	**60.3**	5.1
Human	80.8	100	100

Table 3 F1, HEQ-Q, and HEQ-D results (%) of models and humans on the DoQA test set

Algorithm	F1	HEQ-Q	HEQ-D
BERT	41.4	38.6	4.8
FlowQA	42.8	35.5	5.0
GraphFlow (dynamic graph)	45.3	**41.5**	5.3
GraphFlow (dependency graph)	44.3	39.8	5.0
GraphFlow (hybrid graph)	**46.1**	40.7	**6.8**
Human	86.7	–	–

graph.

In Tables 4 and 5, we experiment on the development dataset of CoQA and QuAC, which demonstrates that the hybrid graph is better than separate dynamic or dependency graphs again. Compared with the dynamic graph, the model with the hybrid graph improves F1 by 1% on QuAC, and 0.1% on CoQA. Besides, it also shows that the constituency graph obtains a better grade than the dependency graph on CoQA.

Table 4 F1 and EM results (%) of different ways to construct the graph on the CoQA development set

Graph type	F1	EM
Dynamic graph	78.1	69.0
Dependency graph	77.8	68.8
Constituency graph	**79.1**	**70.3**
Hybrid graph	78.2	69.6

Table 5 F1, HEQ-Q, and HEQ-D results (%) of different ways to construct the graph on the QuAC development set

Graph type	F1	HEQ-Q	HEQ-D
Dynamic graph	64.8	60.1	**7.0**
Dependency graph	65.3	60.8	6.7
Hybrid graph	**65.8**	**61.5**	5.7

In addition, we explore the influence of different k in the KNN-style approach, which is shown in Fig. 3. The parameter k is a tunable hyperparameter in the KNN-style approach. Fig. 3 shows that it has a role in the performance of the experiment, but the effect is not significant. The best results are obtained when k is set to 15.

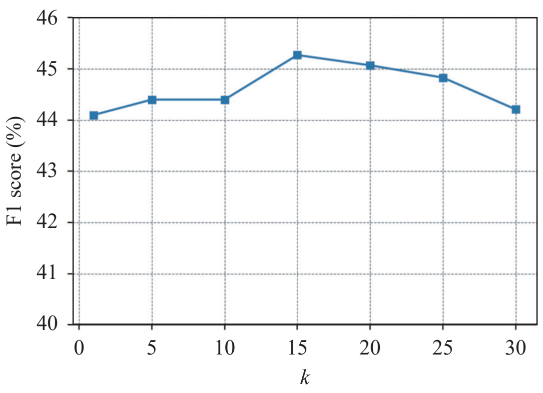


Fig. 3 Influence of different k in the dynamic graph construction section

3.4 Ablation study

We perform an ablation test on our model with the CoQA development dataset by removing several components. The results are shown in Table 6. Here is the explanation of it: –RecurrentConn means to remove all the temporal connections between consecutive context graphs, –RGNN means to remove the RGNN module, –kNN means to remove the kNN-style operation and retain the original weighted adjacency matrix, –PreQues means that no previous answer will be added before the current turn, –PreAns means that no previous answers will be added to the current turn, –PreAnsLoc means that the position of the previous answer will not be marked in the context, and –BERT means to remove pretrained BERT embedding in the encoding layer. We also demonstrate the model′s performance by removing either the entire conversation history in GraphFlow+ (0–His) or one previous turn in GraphFlow+ (1–His).

Table 6 Ablation study results (%) on CoQA development set

	F1
GraphFlow (2–His)	**78.3**
– PreQues	78.2
– PreAns	77.7
– PreAnsLoc	76.6
– BERT	76.0
– RecurrentConn	69.9
– RGNN	68.8
– KNN	69.9
GraphFlow (1–His)	78.2
GraphFlow (0–His)	76.7

3.5 Model analysis

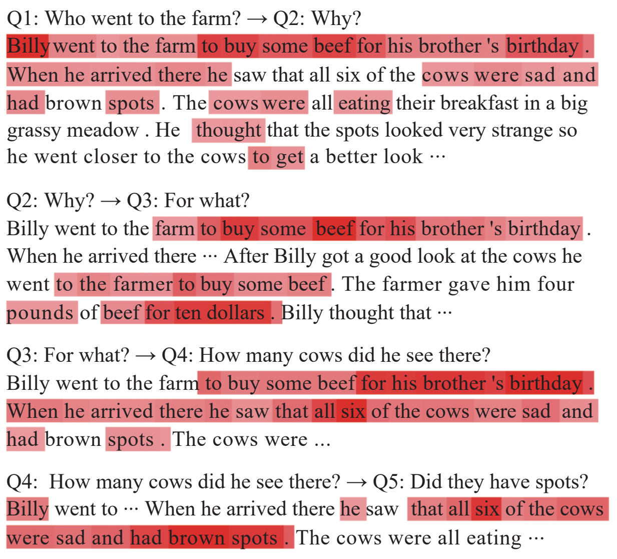
Table 6 illustrates the impact of various model components on the performance of the CoQA development set. The introduction of the proposed RGNN module significantly boosts the F1 score by 7.2%. Both the GNN and the temporal connection part in the RGNN module also contribute to the improvement, indicating the effectiveness of using graph representations for passages and modelling temporal dependencies across a series of contextual graphs. Incorporating conversation history into the current turn enhances the model's performance, and the information from previous answers is more impactful than that from previous questions. Among the methods for utilizing information from previous answers, directly marking the location of previous answers seems to be the most effective. The strong performance of the model when using pretrained BERT embeddings demonstrates the capability of pretrained language models.

Our model outperforms the baseline on most of the major datasets and metrics, which indicates that our model produces more accurate answers in general. While our method performs worse than the baseline on the QuAC dataset in the metric of HEQ-D, which judges the model's performance at the dialogue level. HEQ-D calculates the percentage of dialogues for which the F1 score of the model is higher than the F1 score of human answers for all questions in the dialogue. This indicates that our model is slightly weaker in answering all questions in a conversation well. Moreover, the HEQ-D scores of all models are very low compared to humans, which indicates that the existing models are lacking in this area.

Likewise, experiments on Tables 3–5 investigate the impact of the graph structure. In the three datasets, we find that combining static and dynamic graphs was better than using the two graphs separately. This is because the hybrid graph can learn the structure of both graphs at each turn with the right parameters, which can result in a graph that combines the advantages of both graphs. Moreover, we can find that the constituency graph improves the F1 score by 1% compared to the dynamic graph and 1.3% compared to the dependency graph. This shows that richer information between nodes and edges in the constituency graph helps to improve the performance of the model. However, in the QuAC dataset, the HEQ-D score of the hybrid graph is not as good as that of the dynamic graph or dependency graph, possibly because it needs to balance learning features from both graphs, which may not be stable enough to maintain good performance throughout multiple turns of conversation.

3.6 Interpretability analysis

As in [7], we visually examine the variations in the hidden representations of context words across consecutive turns. To do this, we calculate the cosine similarity between the hidden representations of identical context words at each turn and then highlight words with low cosine similarity scores. Fig. 4 highlights the context words


Q1: Who went to the farm? → Q2: Why?
Billy went to the farm to buy some beef for his brother 's birthday . When he arrived there he saw that all six of the cows were sad and had brown spots . The cows were all eating their breakfast in a big grassy meadow . He thought that the spots looked very strange so he went closer to the cows to get a better look ⋯

Q2: Why? → Q3: For what?
Billy went to the farm to buy some beef for his brother 's birthday . When he arrived there ⋯ After Billy got a good look at the cows he went to the farmer to buy some beef . The farmer gave him four pounds of beef for ten dollars . Billy thought that ⋯

Q3: For what? → Q4: How many cows did he see there?
Billy went to the farm to buy some beef for his brother 's birthday . When he arrived there he saw that all six of the cows were sad and had brown spots . The cows were ...

Q4: How many cows did he see there? → Q5: Did they have spots?
Billy went to ⋯ When he arrived there he saw that all six of the cows were sad and had brown spots . The cows were all eating ⋯

Fig. 4 Highlighted portion of the context shows that GraphFlow+ shifts its focus between consecutive turns of the question (we do not show the full context due to page limitations)

that vary most between successive turns in the conversation of the CoQA dev. set. The hidden representations of context words associated with successive questions change the most and are therefore highlighted the most. It is likely that when the focus changes, the model perceives the context chunks associated with the previous turn as less significant, while those associated with the current turn are more crucial, resulting in updates to the memory in these areas.

4 Related work

4.1 Conversational MC

Conversational MC has achieved great success thanks to the attention mechanism, which can effectively capture the interaction between context and question[26, 27]. However, the problem of how to exploit the conversation histories is not resolved. Many methods have been proposed to utilize conversation history in the literature of conversational MC to exploit conversation history. Among them, Reddy et al.[4, 5] attached all previous questions and answers to the current question. As mentioned in [3], the turn number vector is combined with question embedding and the previous answer locations are combined with context embedding. Nevertheless, these approaches don't use the result of previous reasoning processes at each turn. Huang et al.[7] proposed a novel mechanism IF to allow rich information flow between different turns in the reasoning procedure.

Besides, how to handle abstractive answers remains a challenge in this task. Reddy et al.[4] proposed a hybrid method DrQA+PGNet, which uses a text generator to increase the model. And Yatskar[6] proposed to make a Yes/No judgment, and output an answer according to whether Yes/No was selected. Our work and some other works[5, 7, 28–30] follow a similar idea to solve the challenge.

Previous approaches always treat the passage as a word sequence, while some recent works[8, 9] build the context graph and use the GNN instead of some sequential neural network models. They get a promising result which indicates the high potential of GNNs.

4.2 Graph neural networks

In recent years, graph neural networks (GNNs) have stirred up extensive attention[11, 31–34] in the field of NLP, since there are more people who adapt traditional deep learning techniques to non-euclidean data. GNNs are suitable for modelling relations among elements, and their potential in complex reasoning tasks can be further explored. Recently, GNNs have been successfully used for various question answering QA tasks, which include knowledge base question answering (KBQA)[35], question generation (QG)[36], and machine comprehension (MC)[8, 9].

4.3 Graph construction

For the static graph, most approaches on MC construct a static graph by exploiting prior knowledge in the passage context. These approaches[8, 9, 15, 37, 38] extract entity mentions from the passage as nodes, and use edges that capture different types of relations (e.g., dependency parsing, coreference) to connect the nodes. In our work, we also follow this method to construct the static graph. For dynamic graphs, using attention mechanisms to construct a dynamic graph is still a generic approach in NLP[16, 39, 40], but it is rarely used in MC tasks. Moreover, the hybrid graph[16] can combine different types of graphs to obtain all their advantages, and its potential can be further exploited.

5 Conclusions

We introduce GraphFlow+, a new GNN-based model for conversational machine understanding that uses a novel RGNN for delivering inference outputs throughout a conversation. In addition, we use a range of graph structure learning techniques, including the dynamic graph, constituency graph, and hybrid graph. The hybrid graph combines the advantages of dynamic graphs and static graphs to obtain a balance between the two. In three recently published conversational MC benchmarks, our proposed model achieved competitive results compared to previous approaches. And we also conduct experiments to explore the effectiveness of different types of graphs, and the results show that our novel hybrid graph outperforms both the dynamic graph and the static graph alone. Additionally, the interpretability analysis demonstrates that it provides clear explanations for its predictions.

Acknowledgements

This article is a substantial extension of our earlier work presented at the 29th International Joint Conference on Artificial Intelligence and the 17th Pacific Rim International Conference on Artificial Intelligence (IJCAI-PRICAI-20) in 2020.

Declarations of conflict of interest

The authors declared that they have no conflicts of interest to this work.

References

[1] L. F. Wu, Y. Chen, K. Shen, X. J. Guo, H. N. Gao, S. C. Li, J. Pei, B. Long. Graph neural networks for natural language processing: A survey, [Online], Available: https://arxiv.org/abs/2106.06090, 2021.

[2] P. Rajpurkar, J. Zhang, K. Lopyrev, P. Liang. SQuAD: 100 000+ questions for machine comprehension of text. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, Austin, USA, pp. 2383–2392, 2016. DOI: 10.18653/v1/D16-1264.

[3] E. Choi, H. He, M. Iyyer, M. Yatskar, W. T. Yih, Y. Choi, P. Liang, L. Zettlemoyer. QuAC: Question answering in context. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium, pp. 2174–2184, 2018. DOI: 10.18653/v1/D18-1241.

[4] S. Reddy, D. Q. Chen, C. D. Manning. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 249–266, 2019. DOI: 10.1162/tacl_a_00266.

[5] C. G. Zhu, M. Zeng, X. D. Huang. SDNet: Contextualized attention-based deep network for conversational question answering, [Online], Available: https://arxiv.org/abs/1812.03593, 2018.

[6] M. Yatskar. A qualitative comparison of CoQA, squad 2.0 and QuAC. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Minneapolis, USA, pp. 2318–2323, 2019. DOI: 10.18653/v1/N19-1241.

[7] H. Y. Huang, E. Choi, W. T. Yih. FlowQA: Grasping flow in history for conversational machine comprehension. In *Proceedings of the 7th International Conference on Learning Representations*, New Orleans, USA, 2019.

[8] N. De Cao, W. Aziz, I. Titov. Question answering by reasoning across documents with graph convolutional networks. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Minneapolis, USA, pp. 2306–2317, 2019. DOI: 10.18653/v1/N19-1240.

[9] L. F. Song, Z. G. Wang, M. Yu, Y. Zhang, R. Florian, D. Gildea. Exploring graph-structured passage representation for multi-hop reading comprehension with graph neural networks, [Online], Available: https://arxiv.org/abs/1809.02040, 2018.

[10] Y. Chen, L. F. Wu, M. J. Zaki. GraphFlow: Exploiting

conversation flow with graph neural networks for conversational machine comprehension. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, Yokohama, Japan, pp. 1230–1236, 2020.

[11] L. F. Wu, P. Cui, J. Pei, L. Zhao. *Graph Neural Networks: Foundations, Frontiers, and Applications*, Singapore, Singapore: Springer, 2022. DOI: 10.1007/978-981-16-6054-2.

[12] J. Pennington, R. Socher, C. Manning. GloVe: Global vectors for word representation. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, Doha, Qatar, pp. 1532–1543, 2014. DOI: 10.3115/v1/D14-1162.

[13] J. Devlin, M. W. Chang, K. Lee, K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Minneapolis, USA, pp. 4171–4186, 2019. DOI: 10.18653/v1/N19-1423.

[14] K. Lee, S. Salant, T. Kwiatkowski, A. Parikh, D. Das, J. Berant. Learning recurrent span representations for extractive question answering, [Online], Available: https://arxiv.org/abs/1611.01436, 2016.

[15] K. Xu, L. F. Wu, Z. G. Wang, M. Yu, L. W. Chen, V. Sheinin. Exploiting rich syntactic information for semantic parsing with graph-to-sequence model. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium, pp. 918–924, 2018. DOI: 10.18653/v1/D18-1110.

[16] Y. Chen, L. F. Wu, M. Zaki. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, Vancouver, Canada, pp. 19314–19326, 2020.

[17] S. Hochreiter, J. Schmidhuber. Long short-term memory. *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. DOI: 10.1162/neco.1997.9.8.1735.

[18] Y. J. Li, D. Tarlow, M. Brockschmidt, R. S. Zemel. Gated graph sequence neural networks, [Online], Available: https://arxiv.org/abs/1511.05493, 2015.

[19] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, Doha, Qatar, pp. 1724–1734, 2014. DOI: 10.3115/v1/D14-1179.

[20] W. Wang, M. Yan, C. Wu. Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, Melbourne, Australia, pp. 1705–1714, 2018. DOI: 10.18653/v1/P18-1158.

[21] J. A. Campos, A. Otegi, A. Soroa, J. Deriu, M. Cieliebak, E. Agirre. Conversational QA for FAQs. In *Proceedings of the 33rd Conference on Neural Information Processing Systems*, Vancouver, Canada, 2019.

[22] A. See, P. J. Liu, C. D. Manning. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada, pp. 1073–1083, 2017. DOI: 10.18653/v1/P17-1099.

[23] D. Q. Chen, A. Fisch, J. Weston, A. Bordes. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada, pp. 1870–1879, 2017. DOI: 10.18653/v1/P17-1171.

[24] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, D. McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Baltimore, USA, pp. 55–60, 2014. DOI: 10.3115/v1/P14-5010.

[25] D. P. Kingma, J. Ba. Adam: A method for stochastic optimization, [Online], Available: https://arxiv.org/abs/1412.6980, 2014.

[26] M. Seo, A. Kembhavi, A. Farhadi, H. Hajishirzi. Bidirectional attention flow for machine comprehension. In *Proceedings of the 5th International Conference on Learning Representations*, Toulon, France, 2017.

[27] C. M. Xiong, V. Zhong, R. Socher. Dynamic coattention networks for question answering. In *Proceedings of the 5th International Conference on Learning Representations*, Toulon, France, 2017.

[28] Y. Ju, F. B. Zhao, S. J. Chen, B. W. Zheng, X. F. Yang, Y. F. Liu. Technical report on conversational question answering, [Online], Available: https://arxiv.org/abs/1909.10772, 2019.

[29] C. Qu, L. Yang, M. H. Qiu, Y. F. Zhang, C. Chen, W. B. Croft, M. Iyyer. Attentive history selection for conversational question answering. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Managemen*, Beijing, China, pp. 1391–1400, 2019. DOI: 10.1145/3357384.3357905.

[30] Y. T. Yeh, Y. N. Chen. FlowDelta: Modeling flow information gain in reasoning for conversational machine comprehension. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, Hong Kong, China, pp. 86–90, 2019. DOI: 10.18653/v1/D19-5812.

[31] K. Xu, L. F. Wu, Z. G. Wang, V. Sheinin. Graph2Seq: Graph to sequence learning with attention-based neural networks, [Online], Available: https://arxiv.org/abs/1804.00823, 2018.

[32] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, G. E. Dahl. Neural message passing for Quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, Sydney, Australia, pp. 1263–1272, 2017.

[33] W. L. Hamilton, R. Ying, J. Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Long Beach, USA, pp. 1025–1035, 2017.

[34] T. N. Kipf, M. Welling. Semi-supervised classification with graph convolutional networks, [Online], Available: https://arxiv.org/abs/1609.02907, 2016.

[35] H. T. Sun, B. Dhingra, M. Zaheer, K. Mazaitis, R. Salakhutdinov, W. Cohen. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium, pp. 4231–4242, 2018. DOI: 10.18653/v1/D18-1455.

[36] Y. Chen, L. F. Wu, M. J. Zaki. Toward subgraph guided knowledge graph question generation with graph neural networks, [Online], Available: https://arxiv.org/abs/2004.06015, 2020.

[37] Q. Ran, Y. K. Lin, P. Li, J. Zhou, Z. Y. Liu. NumNet: Machine reading comprehension with numerical reasoning. In

Proceedings of Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, Hong Kong, China, pp. 2474–2484, 2019. DOI: 10.18653/v1/D19-1251.

[38] C. Zheng, P. Kordjamshidi. SRLGRN: Semantic role labeling graph reasoning network. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, pp. 8881–8891, 2020. DOI: 10.18653/v1/2020.emnlp-main.714.

[39] P. F. Liu, S. C. Chang, X. J. Huang, J. Tang, J. C. K. Cheung. Contextualized non-local neural networks for sequence learning. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence and the 31st Innovative Applications of Artificial Intelligence Conference and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence*, Honolulu, USA, pp. 830, 2018. DOI: 10.1609/aaai.v33i01.33016762.

[40] Y. Chen, L. F. Wu, M. J. Zaki. Reinforcement learning based graph-to-sequence model for natural question generation. In *Proceedings of the 8th International Conference on Learning Representations*, Addis Ababa, Ethiopia, 2020.

Jing Hu received the B. Sc. degree in software engineer from Central China Normal University, China in 2023. Currently, she is a master student in computer technology at Tsinghua Shenzhen International Graduate School, Tsinghua University, China.

Her research interests include natural language processing, pretrained language models and code summarization.

E-mail: cminus@mails.ccnu.edu.cn

ORCID iD: 0000-0002-4634-9510

Lingfei Wu received the Ph. D. degree in computer science from College of William and Mary, USA in 2016. Currently, he is an engineering manager in the Content and Knowledge Graph Group at Pinterest, USA. He has published one book (in GNNs) and more than 100 top-ranked AI/ML/NLP conferences and journal papers, including but not limited to NeurIPS, ICML, ICLR, KDD, ACL, EMNLP, NAACL, IJCAI, and AAAI. He is also a co-inventor of more than 40 filed US patents. Because of the commercial value of his patents, he received several invention achievement awards and was appointed as IBM Master Inventors, class of 2020. He was the recipient of the Best Paper Award and Best Student Paper Award of several conferences such as IEEE ICC19, AAAI workshop on DLGMA20 and KDD workshop on DLG19.

His research interests include the intersection of machine learning (deep learning), representation learning, and natural language processing, with a particular emphasis on the fast-growing subjects of graph neural networks and its extensions on new application domains.

E-mail: teddy.lfwu@gmail.com (Corresponding author)

ORCID iD: 0000-0002-3660-651X

Yu Chen received the Ph. D. degree in computer science from Rensselaer Polytechnic Institute, USA in 2020. Currently, he is a senior research scientist at Meta AI, USA.

His research interests include the intersection of machine learning (deep learning) and natural language processing, with a particular emphasis on the fast-growing field of graph neural networks and their applications in various domains.

E-mail: hugochan2013@gmail.com

Po Hu received the Ph. D. degree in computer software and theory from Wuhan University, China in 2013. He was a visiting scholar at Hong Kong Baptist University, China. Currently, he is an associate professor of computer science at Central China Normal University, China. He has published over 30 papers in prestigious journals and prominent conferences (e.g., IJCAI, ACL, COLING). He has served as the program committee/journal reviewer in major AI and NLP conferences (e.g., AAAI, ACL, EMNLP, TKDE).

His research interests include natural language processing, knowledge engineering and computational social science.

E-mail: phu@mail.ccnu.edu.cn

Mohammed J. Zaki received the Ph. D. degree in computer science from University of Rochester, USA in 1998. Currently, he is a professor and department head of computer science at Rensselaer Polytechnic Institute, USA.

His research interests include novel data mining and machine learning techniques, particularly for learning from graph structured and textual data, with applications in bioinformatics, personal health and financial analytics.

E-mail: zaki@cs.rpi.edu