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

Yu Chen[1] , **Lingfei Wu**[2] * and **Mohammed J. Zaki**[1]
[1]Rensselaer Polytechnic Institute
[2]IBM Research
cheny39@rpi.edu, lwu@email.wm.edu, zaki@cs.rpi.edu

Abstract

Conversational machine comprehension (MC) has proven significantly more challenging compared to traditional MC since it requires better utilization of conversation history. However, most existing approaches do not effectively capture conversation history and thus have trouble handling questions involving coreference or ellipsis. Moreover, when reasoning over passage text, most of them simply treat it as a word sequence without exploring rich semantic relationships among words. In this paper, we first propose a simple yet effective graph structure learning technique to dynamically construct a question and conversation history aware context graph at each conversation turn. Then we propose a novel Recurrent Graph Neural Network, and based on that, we introduce a flow mechanism to model the temporal dependencies in a sequence of context graphs. The proposed GRAPHFLOW model can effectively capture conversational flow in a dialog, and shows competitive performance compared to existing state-of-the-art methods on CoQA, QuAC and DoQA benchmarks. In addition, visualization experiments show that our proposed model can offer good interpretability for the reasoning process.

1 Introduction

Recent years have observed a surge of interest in conversational machine comprehension (MC). Unlike the setting of traditional MC that requires answering a single question given a passage, the conversational MC task is to answer a question in a conversation given a passage and all previous questions and answers. Despite the success existing works have achieved on MC (e.g., SQuAD [Rajpurkar *et al.*, 2016]), conversational MC has proven significantly more challenging. We highlight two major challenges here. First, the focus usually shifts as a conversation progresses [Reddy *et al.*, 2018; Choi *et al.*, 2018]. Second, many questions refer back to conversation history via coreference or ellipsis. Therefore, without fully utilizing conversation history (i.e., previous questions and/or answers), one can not understand a question correctly. In this work, we model the concept of *conversation flow* as a sequence of latent states associated with these shifts of focus in a conversation.

To cope with the above challenges, many methods have been proposed to utilize conversation history. Most of them simply prepend the conversation history to a current question [Reddy *et al.*, 2018; Zhu *et al.*, 2018] or add previous answer locations to a passage [Choi *et al.*, 2018; Yatskar, 2018], and then treat the task as single-turn MC without explicitly modeling conversation flow. Huang et al. [2018] assumed that hidden representations generated during previous reasoning processes potentially capture important information for answering a current question. In order to model conversation flow, they proposed an *Integration-Flow* (IF) mechanism to first perform sequential reasoning over passage words in parallel for each turn, and then refine the reasoning results sequentially across different turns, in parallel of passage words.

However, the IF mechanism has several limitations when reasoning over a sequence of passages for answer seeking. First of all, the strategy of interweaving two processing directions (i.e., in passage words and in question turns) is not quite effective. Because in the IF mechanism, the results of previous reasoning processes are not incorporated into the current reasoning process immediately. Instead, all reasoning processes over passage words are conducted in parallel. As a result, the reasoning performance at each turn is not improved by the outcome of previous reasoning processes. To alleviate this issue, they have to refine the reasoning results sequentially across different turns and use stacked IF layers to interweave two processing directions multiple times. Second, following most previous methods, when reasoning over passage text, they simply treat it as a word sequence without exploring the rich semantic relationships among words. Recent works on multi-hop MC [De Cao *et al.*, 2018; Song *et al.*, 2018] have shown the advantages of applying a Graph Neural Network (GNN) to process a passage graph over simply processing a word sequence using a Recurrent Neural Network (RNN).

To better capture conversation flow and address the above issues, in this work, we propose GRAPHFLOW, a GNN based model for conversational MC. We first propose a simple yet effective graph structure learning technique to dynamically construct a question and conversation history aware context graph at each turn that consists of each word as a node.

*Corresponding author.

Then we propose a novel Recurrent Graph Neural Network (RGNN), and based on that, we introduce a flow mechanism to model the temporal dependencies in a sequence of context graphs. Answers are finally predicted based on the matching score of the question embedding and the context graph embedding at each turn.

We highlight our contributions as follows:

- We propose a novel GNN based model, namely GRAPHFLOW, for conversational MC which captures conversational flow in a dialog.
- We dynamically construct a question and conversation history aware context graph at each turn, and propose a novel Recurrent Graph Neural Network based flow mechanism to process a sequence of context graphs.
- On three public benchmarks (i.e., CoQA, QuAC and DoQA), our model shows competitive performance compared to existing state-of-the-art methods. In addition, visualization experiments shows that our model can offer good interpretability for the reasoning process.

2 Related Work

2.1 Conversational MC

One big challenge of Conversational MC is how to effectively utilize conversation history. [Reddy *et al.*, 2018; Zhu *et al.*, 2018] concatenated previous questions and answers to the current question. Choi et al. [2018] concatenated a feature vector encoding the turn number to the question word embedding and a feature vector encoding previous N answer locations to the context embeddings. However, these methods ignore previous reasoning processes performed by the model when reasoning at the current turn. Huang et al. [2018] proposed the idea of *Integration-Flow* (IF) to allow rich information in the reasoning process to flow through a conversation. To better model conversation flow, in this work, we propose a novel GNN based flow mechanism to sequentially process a sequence of context graphs.

Another challenge of this task is how to handle abstractive answers. Reddy et al. [2018] propose a hybrid method DrQA+PGNet, which augments a traditional extractive reading comprehension model with a text generator. Yatskar [2018] propose to first make a Yes/No decision, and output an answer span only if Yes/No was not selected. Recent work [Huang *et al.*, 2018; Zhu *et al.*, 2018; Yeh and Chen, 2019; Qu *et al.*, 2019; Ju *et al.*, 2019] as well as our work in this paper follows a similar idea to handle abstractive answers.

When processing passage text in MC, most existing methods treat it as a word sequence. Recently, promising results have been achieved by applying a GNN to process a passage graph [De Cao *et al.*, 2018; Song *et al.*, 2018].

2.2 Graph Neural Networks

Over the past few years, graph neural networks (GNNs) [Kipf and Welling, 2016; Gilmer *et al.*, 2017; Hamilton *et al.*, 2017; Xu *et al.*, 2018a] have drawn increasing attention. Recently, GNNs have been applied to various question answering tasks including knowledge base question answering (KBQA) [Sun *et al.*, 2018], question generation [Chen *et al.*, 2020], and MC [De Cao *et al.*, 2018; Song *et al.*, 2018], and have shown advantages over traditional approaches. For tasks where the graph structure is unknown, linguistic features (e.g., dependency parsing, coreferences) [Xu *et al.*, 2018b; De Cao *et al.*, 2018; Song *et al.*, 2018] or attention-based mechanisms [Liu *et al.*, 2018; Chen *et al.*, 2019b; Chen *et al.*, 2019a] are usually used to construct a static or dynamic graph containing words or entity mentions as nodes.

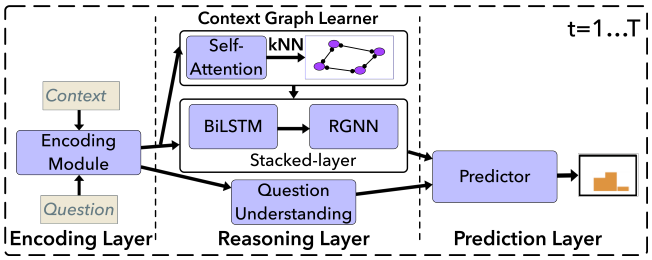

Figure 1: Overall architecture of the proposed model.

3 The GraphFlow Approach

The task of conversational MC is to answer a natural language question given the context and conversation history. Let us denote C as the context which consists of a word sequence $\{c_1, c_2, ..., c_m\}$ and $Q^{(i)}$ as the question at the i-th turn which consists of a word sequence $\{q_1^{(i)}, q_2^{(i)}, ..., q_n^{(i)}\}$. And there are totally T turns in a conversation.

As shown in Fig. 1, our proposed GRAPHFLOW model consists of *Encoding Layer*, *Reasoning Layer* and *Prediction Layer*. The *Encoding Layer* encodes conversation history and context that aligns question information. The *Reasoning Layer* dynamically constructs a question and conversation history aware context graph at each turn, and then applies a flow mechanism to process a sequence of context graphs. The *Prediction Layer* predicts the answers based on the matching score of the question embedding and the context graph embedding. The details of these modules are given next.

3.1 Encoding Layer

We apply an effective encoding layer to encode the context and the question, which additionally exploits conversation history and interactions between them.

Linguistic features. For context word c_j, we encode linguistic features into a vector $f_{\text{ling}}(c_j^{(i)})$ concatenating POS (part-of-speech), NER (named entity recognition) and exact matching (which indicates whether c_j appears in $Q^{(i)}$) embeddings.

Pretrained word embeddings. We use 300-dim GloVe [Pennington *et al.*, 2014] embeddings and 1024-dim BERT [Devlin *et al.*, 2018] embeddings to embed each word in the context and the question. Compared to GloVe, BERT better utilizes contextual information when embedding words.

Aligned question embeddings. Exact matching matches words on the surface form; we further apply an attention mechanism to learn soft alignment between context words and question words. Since this soft alignment operation is

conducted in parallel at each turn, for the sake of simplicity, we omit the turn index i when formulating the alignment operation. Following Lee et al. [2016], for context word c_j at each turn, we incorporate an aligned question embedding

$$f_{align}(c_j) = \sum_k \beta_{j,k} \mathbf{g}_k^Q \tag{1}$$

where $\mathbf{g}_k^Q$ is the GloVe embedding of the k-th question word q_k and $\beta_{j,k}$ is an attention score between context word c_j and question word q_k. The attention score $\beta_{j,k}$ is computed by

$$\beta_{j,k} \propto \exp(\text{ReLU}(\mathbf{W}\mathbf{g}_j^C)^T \text{ReLU}(\mathbf{W}\mathbf{g}_k^Q)) \tag{2}$$

where $\mathbf{W}$ is a $d \times 300$ trainable weight with d being the hidden state size, and $\mathbf{g}_j^C$ is the GloVe embedding of context word c_j. To simplify notation, we denote the above attention mechanism as $\text{Align}(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$, meaning that an attention matrix is computed between two sets of vectors $\mathbf{X}$ and $\mathbf{Y}$, which is later used to get a linear combination of vector set $\mathbf{Z}$. Hence we can reformulate the above alignment as

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

Conversation history. Following Choi et al. [2018], we concatenate a feature vector $f_{\text{ans}}(c_j^{(i)})$ encoding previous N answer locations to context word embeddings. Preliminary experiments showed that it is helpful to also prepend previous N question-answer pairs to a current question. In addition, to each word vector in an augmented question, we concatenate a turn marker embedding $f_{\text{turn}}(q_k^{(i)})$ indicating which turn the word belongs to.

In summary, at the i-th turn in a conversation, each context word c_j is encoded by a vector $\mathbf{w}_{c_j}^{(i)}$ which is a concatenation of linguistic vector $f_{\text{ling}}(c_j^{(i)})$, word embeddings (i.e., $\mathbf{g}_j^C$ and BERT_j^C), aligned vector $f_{\text{align}}(c_j^{(i)})$ and answer vector $f_{\text{ans}}(c_j^{(i)})$. And each question word $q_k^{(i)}$ is encoded by a vector $\mathbf{w}_{q_k}^{(i)}$ which is a concatenation of word embeddings (i.e., $\mathbf{g}_k^{Q^{(i)}}$ and $\text{BERT}_k^{Q^{(i)}}$) and turn marker vector $f_{\text{turn}}(q_k^{(i)})$. We denote $\mathbf{W}_C^{(i)}$ and $\mathbf{W}_Q^{(i)}$ as a sequence of context word vectors $\mathbf{w}_{c_j}^{(i)}$ and question word vectors $\mathbf{w}_{q_k}^{(i)}$, respectively.

3.2 Reasoning Layer

When performing reasoning over context, unlike most previous methods that regard context as a word sequence, we opt to treat context as a "graph" of words that captures rich semantic relationships among words, and apply a Recurrent Graph Neural Network to process a sequence of context graphs.

Question Understanding

For a question $Q^{(i)}$, we apply a bidirectional LSTM [Hochreiter and Schmidhuber, 1997] to the question embeddings $\mathbf{W}_Q^{(i)}$ to obtain contextualized embeddings $\mathbf{Q}^{(i)} \in \mathbb{R}^{d \times n}$.

$$\mathbf{Q}^{(i)} = \mathbf{q}_1^{(i)}, ..., \mathbf{q}_n^{(i)} = \text{BiLSTM}(\mathbf{W}_Q^{(i)}) \tag{4}$$

And the question is then represented as a weighted sum of question word vectors via a self attention mechanism,

$$\tilde{\mathbf{q}}^{(i)} = \sum_k a_k^{(i)} \mathbf{q}_k^{(i)}, \text{ where } a_k^{(i)} \propto \exp(\mathbf{w}^T \mathbf{q}_k^{(i)}) \tag{5}$$

where $\mathbf{w}$ is a d-dim trainable weight.

Finally, to capture the dependency among question history, we encode the sequence of questions with a LSTM to generate history-aware question vectors.

$$\mathbf{p}^{(1)}, ..., \mathbf{p}^{(T)} = \text{LSTM}(\tilde{\mathbf{q}}^{(1)}, ..., \tilde{\mathbf{q}}^{(T)}) \tag{6}$$

The output hidden states of the LSTM network $\mathbf{p}^{(1)}, ..., \mathbf{p}^{(T)}$ will be used for predicting answers.

Context Graph Learning

The intrinsic context graph structure is unfortunately unknown. Moreover, the context graph structure might vary across different turns by considering the changes of questions and conversation history. Most existing applications of GNNs [Xu *et al.*, 2018b; De Cao *et al.*, 2018; Song *et al.*, 2018] use ground-truth or manually constructed graphs which have some limitations. First, the ground-truth graphs are not always available. Second, errors in manual construction process can be propagated to subsequent modules. Unlike previous methods, we automatically construct graphs from raw context, which are combined with the rest of the system to make the whole learning system end-to-end trainable. We dynamically build a question and conversation history aware context graph to model semantic relationships among context words at each turn.

Specifically, we first apply an attention mechanism to the context representations $\mathbf{W}_C^{(i)}$ (which additionally incorporate both question information and conversation history) at the i-th turn to compute an attention matrix $\mathbf{A}^{(i)}$, serving as a weighted adjacency matrix for the context graph, defined as,

$$\mathbf{A}^{(i)} = (\mathbf{W}_C^{(i)} \odot \mathbf{u})^T \mathbf{W}_C^{(i)} \tag{7}$$

where $\odot$ denotes element-wise multiplication, and $\mathbf{u}$ is a non-negative d_c-dim trainable weight vector which learns to highlight different dimensions of $\mathbf{w}_{c_j}^{(i)}$ whose dimension is d_c.

Considering that a fully connected context graph is not only computationally expensive but also might introduce noise (i.e., unimportant edges), a simple kNN-style graph sparsification operation is applied to select the most important edges from the fully connected graph, resulting in a sparse graph. To be concrete, given a learned attention matrix $\mathbf{A}^{(i)}$, we only keep the K nearest neighbors (including itself) as well as the associated attention scores (i.e., the remaining attentions scores are masked off) for each context node. We then apply a softmax function to these selected adjacency matrix elements to get a normalized adjacency matrix.

$$\tilde{\mathbf{A}}^{(i)} = \text{softmax}(\text{topk}(\mathbf{A}^{(i)})) \tag{8}$$

Note that the supervision signal is still able to back-propagate through the kNN-style graph sparsification module since the K nearest attention scores are kept and used to compute the weights of the final normalized adjacency matrix.

Context Graph Reasoning

When reasoning over a sequence of context graphs, we want to consider not only the relationships among graph nodes, but also the sequential dependencies among graphs. Especially

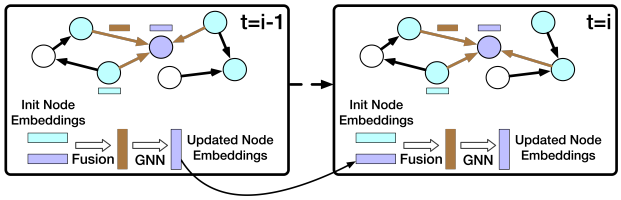

Figure 2: Architecture of the proposed Recurrent Graph Neural Network for processing a sequence of context graphs.

for the conversational MC task, we hope the results of previous reasoning processes can be incorporated into the current reasoning process since they potentially capture important information for answering the current question.

Therefore, we propose a novel *Recurrent Graph Neural Network* (RGNN) to process a sequence of graphs, as shown in Fig. 2. As we advance in a sequence of graphs, we process each graph using a shared GNN cell and the GNN output will be used when processing the next graph. One can think that it is analogous to an RNN-style structure where the main difference is that each element in a sequence is not a data point, but instead a graph. Our RGNN module combines the advantages of RNNs which are good at sequential learning (i.e., modeling sequential data), and GNNs which are good at relational reasoning (i.e., modeling graph-structured data).

The computational details of RGNN are as follows. Let us denote $\mathbf{C}^{(i)}$ as the initial context node embedding at the i-th turn. Before we apply a GNN to the context graph $\mathcal{G}^{(i)}$, we update its node embeddings by fusing both the original node information $\mathbf{C}^{(i)}$ and the updated node information $\bar{\mathbf{C}}^{(i-1)}$ computed by a parameter-sharing GNN at the $(i-1)$-th turn via a fusion function,

$$\bar{\mathbf{C}}^{(i)} = \text{GNN}(\text{Fuse}(\mathbf{C}^{(i)}, \bar{\mathbf{C}}^{(i-1)}), \tilde{\mathbf{A}}^{(i)}) \tag{9}$$

where we set $\bar{\mathbf{C}}^{(0)} = \mathbf{C}^0$ as we do not incorporate any historical information at the first turn. The fusion function is designed as a gated sum of two information sources,

$$\begin{aligned} \text{Fuse}(\mathbf{a}, \mathbf{b}) &= \mathbf{z} * \mathbf{a} + (1 - \mathbf{z}) * \mathbf{b} \\ \mathbf{z} &= \sigma(\mathbf{W}_z[\mathbf{a}; \mathbf{b}; \mathbf{a} * \mathbf{b}; \mathbf{a} - \mathbf{b}] + \mathbf{b}_z) \end{aligned} \tag{10}$$

where σ is a sigmoid function and $\mathbf{z}$ is a gating vector. As a result, the graph node embedding outputs of the reasoning process at the previous turn are used as a starting state when reasoning at the current turn.

We use Gated Graph Neural Networks (GGNN) [Li *et al.*, 2015] as our GNN cell, but the framework is agnostic to the particular choice of GNN cell. In GGNN we do multi-hop message passing through a graph to capture long-range dependency where the same set of network parameters are shared at every hop of computation. At each hop of computation, for every graph node, we compute an aggregation vector as a weighted average of all its neighboring node embeddings where the weights come from the normalized adjacency matrices $\tilde{\mathbf{A}}^{(i)}$. Then, a Gated Recurrent Unit (GRU) [Cho *et al.*, 2014] is used to update node embeddings by incorporating the aggregation vectors. We use the updated node embeddings at the last hop as the final node embeddings.

To simplify notation, we denote the above RGNN module as $\bar{\mathbf{C}}^{(i)} = \text{RGNN}(\mathbf{C}^{(i)}, \tilde{\mathbf{A}}^{(i)}),\ i = 1, \ldots, T$ which takes as input a sequence of graph node embeddings $\{\mathbf{C}^{(i)}\}_{i=1}^T$ as well as a sequence of the normalized adjacency matrices $\{\tilde{\mathbf{A}}^{(i)}\}_{i=1}^T$, and outputs a sequence of updated graph node embeddings $\{\bar{\mathbf{C}}^{(i)}\}_{i=1}^T$.

While a GNN is responsible for modeling global interactions among context words, modeling local interactions between consecutive context words is also important for the task. Therefore, before feeding the context word representations to a GNN, we first apply a BiLSTM to encode local dependency, that is, $\mathbf{C}^{(i)} = \text{BiLSTM}(\mathbf{W}_C^{(i)})$, and then use the output $\mathbf{C}^{(i)}$ as the initial context node embedding.

Inspired by recent work [Wang *et al.*, 2018] on modeling the context with different levels of granularity, we choose to apply stacked RGNN layers where one RGNN layer is applied on low level representations of the context and the second RGNN layer is applied on high level representations of the context. The output of the second RGNN layer $\{\tilde{\mathbf{C}}^{(i)}\}_{i=1}^T$ is the final context representations.

$$\begin{aligned} \mathbf{H}_C^{(i)} &= [\bar{\mathbf{C}}^{(i)}; \mathbf{g}^C; \text{BERT}^C] \\ \mathbf{H}_Q^{(i)} &= [\mathbf{Q}^{(i)}; \mathbf{g}^{Q^{(i)}}; \text{BERT}^{Q^{(i)}}] \\ f_{\text{align}}^2(C^{(i)}) &= \text{Align}(\mathbf{H}_C^{(i)}, \mathbf{H}_Q^{(i)}, \mathbf{Q}^{(i)}] \\ \hat{\mathbf{C}}^{(i)} &= \text{BiLSTM}([\bar{\mathbf{C}}^{(i)}; f_{\text{align}}^2(C^{(i)})]) \\ \tilde{\mathbf{C}}^{(i)} &= \text{RGNN}(\hat{\mathbf{C}}^{(i)}, \tilde{\mathbf{A}}^{(i)}),\ i = 1, \ldots, T \end{aligned} \tag{11}$$

3.3 Prediction Layer

We predict answer spans by computing the start and end probabilities of the j-th context word for the i-th question. For the sake of simplicity, we omit the turn index i when formulating the prediction layer. The start probability P_j^S is calculated by,

$$P_j^S \propto \exp(\tilde{\mathbf{c}}_j^T \mathbf{W}_S \mathbf{p}) \tag{12}$$

where $\mathbf{W}_S$ is a $d \times d$ trainable weight and $\mathbf{p}$ (turn index omitted) is the question representation obtained in Eq. (6). Next, $\mathbf{p}$ is passed to a GRU cell by incorporating context summary and converted to $\tilde{\mathbf{p}}$.

$$\tilde{\mathbf{p}} = \text{GRU}(\mathbf{p}, \sum_j P_j^S \tilde{\mathbf{c}}_j) \tag{13}$$

Then, the end probability P_j^E is calculated by,

$$P_j^E \propto \exp(\tilde{\mathbf{c}}_j^T \mathbf{W}_E \tilde{\mathbf{p}}) \tag{14}$$

where $\mathbf{W}_E$ is a $d \times d$ trainable weight.

We apply an answer type classifier to handle unanswerable questions and questions whose answers are not text spans in the context. The probability of the answer type (e.g., "unknown", "yes" and "no") is calculated as follows,

$$P^C = \sigma(f_c(\mathbf{p})[f_{\text{mean}}(\tilde{\mathbf{C}}); f_{\text{max}}(\tilde{\mathbf{C}})]^T) \tag{15}$$

where f_c is a dense layer which maps a d-dim vector to a (num_class $\times$ $2d$)-dim vector. Further, σ is a sigmoid function for binary classification and a softmax function for multi-class classification. $f_{\text{mean}}(.)$ and $f_{\text{max}}(.)$ denote the average pooling and max pooling operations, respectively.

	Child.	Liter.	Mid-High.	News	Wiki	Reddit	Science	Overall
PGNet [See *et al.*, 2017]	49.0	43.3	47.5	47.5	45.1	38.6	38.1	44.1
DrQA [Chen *et al.*, 2017]	46.7	53.9	54.1	57.8	59.4	45.0	51.0	52.6
DrQA+PGNet [Reddy *et al.*, 2018]	64.2	63.7	67.1	68.3	71.4	57.8	63.1	65.1
BiDAF++ [Yatskar, 2018]	66.5	65.7	70.2	71.6	72.6	60.8	67.1	67.8
FLOWQA [Huang *et al.*, 2018]	73.7	71.6	76.8	79.0	80.2	67.8	76.1	75.0
Flow [Unpublished]	–	–	–	–	–	–	–	75.8
SDNet [Zhu *et al.*, 2018]	75.4	73.9	77.1	**80.3**	**83.1**	69.8	76.8	76.6
GRAPHFLOW	**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 1: Model and human performance (% in F1 score) on CoQA test set.

3.4 Training and Inference

The training objective for the i-th turn is defined as the cross entropy loss of both text span prediction (if the question requires it) and answer type prediction where the turn index i is omitted for the sake of simplicity,

$$\mathcal{L} = -I^S(log(P_s^S) + log(P_e^E)) + \log P_t^C \qquad (16)$$

where I^S indicates whether the question requires answer span prediction, s and e are the ground-truth start and end positions of the span, and t indicates the ground-truth answer type.

During inference, we first use P^C to predict whether the question requires text span prediction. If yes, we predict the span to be $\hat{s}, \hat{e}$ with maximum $P_{\hat{s}}^S, P_{\hat{e}}^E$ subject to certain maximum span length threshold.

4 Experiments

In this section, we conduct an extensive evaluation of our proposed model against state-of-the-art conversational MC models. We use three popular benchmarks as described below. The implementation of our model is publicly available at https://github.com/hugochan/GraphFlow.

4.1 Datasets, Baselines and Evaluation Metrics

CoQA [Reddy *et al.*, 2018] contains 127k questions with answers, obtained from 8k conversations. Answers are in free-form and hence are not necessarily text spans from context. The average length of questions is only 5.5 words. The average number of turns per dialog is 15.2. QuAC [Choi *et al.*, 2018] contains 98k questions with answers, obtained from 13k conversations. All the answers are text spans from context. The average length of questions is 6.5 and there are on average 7.2 questions per dialog. DoQA [Campos *et al.*, 2019] contains 7.3k questions with answers, obtained from 1.6k conversations in the cooking domain. Similar to CoQA, 31.3% of the answers are not directly extracted from context.

We compare our method with the following baselines: PGNet [See *et al.*, 2017], DrQA [Chen *et al.*, 2017], DrQA+PGNet [Reddy *et al.*, 2018], BiDAF++ [Yatskar, 2018], FLOWQA [Huang *et al.*, 2018], SDNet [Zhu *et al.*, 2018], BERT [Devlin *et al.*, 2018] and Flow (unpublished).

Following previous works [Huang *et al.*, 2018; Zhu *et al.*, 2018], we use an extractive approach with answer type classifiers on all benchmarks. The main evaluation metric is F1 score which is the harmonic mean of precision and recall at word level between the predication and ground truth. In addition, for QuAC and DoQA, the Human Equivalence Score (HEQ) is used to judge whether a system performs as well as an average human. HEQ-Q and HEQ-D are model accuracies at question level and dialog level. Please refer to [Reddy *et al.*, 2018; Choi *et al.*, 2018] for details of these metrics.

	F1	HEQ-Q	HEQ-D
BiDAF++	60.1	54.8	4.0
FLOWQA	64.1	59.6	**5.8**
GRAPHFLOW	**64.9**	**60.3**	5.1
Human	80.8	100	100

Table 2: Model and human performance (in %) on QuAC test set.

4.2 Model Settings

The embedding sizes of POS, NER, exact matching and turn marker embeddings are set to 12, 8, 3 and 3, respectively. Following Zhu et al. [2018], we pre-compute BERT embeddings for each word using a weighted sum of BERT layer outputs. The size of all hidden layers is set to 300. When constructing context graphs, the neighborhood size is set to 10. The number of GNN hops is set to 5 for CoQA and DoQA, and 3 for QuAC. During training, we apply dropout after embedding layers (0.3 for GloVe and 0.4 for BERT) and RNN layers (0.3 for all). We use Adamax [Kingma and Ba, 2014] as the optimizer and the learning rate is set to 0.001. We batch over dialogs and the batch size is set to 1. When augmenting the current turn with conversation history, we only consider the previous two turns. All these hyper-parameters are tuned on the development set.

4.3 Experimental Results

As shown in Table 1, Table 2, and Table 3, our model outperforms or achieves competitive performance compared with various state-of-the-art baselines. Compared with FLOWQA which is also based on the flow idea, our model improves F1 by 2.3% on CoQA, 0.8% on QuAC and 2.5% on DoQA, which demonstrates the superiority of our RGNN based flow mechanism over the IF mechanism. Compared with SDNet which relies on sophisticated inter-attention and self-attention mechanisms, our model improves F1 by 0.7% on CoQA.

	F1	HEQ-Q	HEQ-D
BERT	41.4	38.6	4.8
FLOWQA	42.8	35.5	5.0
GRAPHFLOW	**45.3**	**41.5**	**5.3**
Human	86.7	–	–

Table 3: Model and human performance (in %) on DoQA test 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

Table 4: Ablation study (in %) on CoQA dev. set.

4.4 Ablation Study and Model Analysis

We conduct an extensive ablation study to further investigate the performance impact of different components in our model. Here we briefly describe ablated systems: – RecurrentConn removes temporal connections between consecutive context graphs, – RGNN removes the RGNN module, – kNN removes the kNN-style graph sparsification operation, – PreQues does not prepend previous questions to the current turn, – PreAns does not prepend previous answers to the current turn, – PreAnsLoc does not mark previous answer locations in the context, and – BERT removes pretrained BERT embeddings. We also show the model performance with no conversation history GRAPHFLOW (0-His) or one previous turn of the conversation history GRAPHFLOW (1-His).

Table 4 shows the contributions of the above components on the CoQA development set. Our proposed RGNN module contributes significantly to the model performance (i.e., improves F1 score by 7.2%). In addition, within the RGNN module, both the GNN part (i.e., 1.1% F1) and the temporal connection part (i.e., 6.1% F1) contribute to the results. This verifies the effectiveness of representing a passage as a graph and modeling the temporal dependencies in a sequence of context graphs. The kNN-style graph sparsification operation also contributes significantly to the model performance. We notice that explicitly adding conversation history to the current turn helps the model performance. We can see that the previous answer information is more crucial than the previous question information. And among many ways to use the previous answer information, directly marking previous answer locations seems to be the most effective. Last but not least, we find that the pretrained BERT embedding has significant impact on the performance, which demonstrates the power of large-scale pretrained language models.

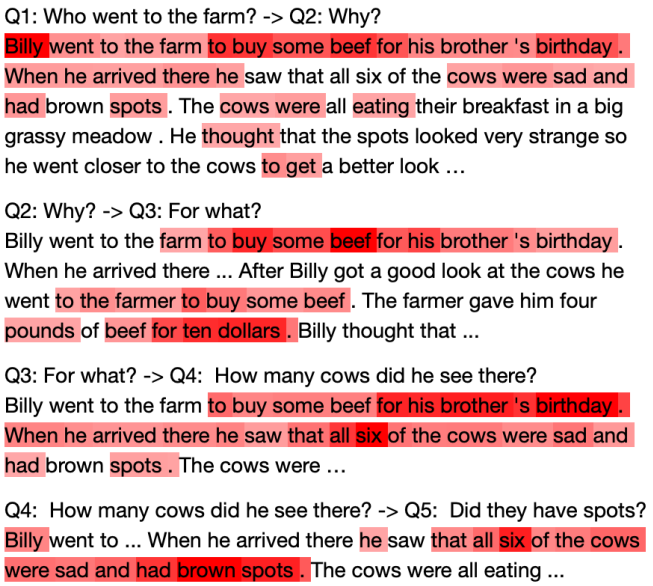

Figure 3: The highlighted part of the context indicates GraphFlow's focus shifts between consecutive question turns.

4.5 Interpretability Analysis

Following Huang et al. [2018], we visualize the changes of hidden representations of context words between consecutive turns. Specifically, we compute cosine similarity of hidden representations of the same context words at consecutive turns, and then highlight the words that have small cosine similarity scores (i.e., change more significantly). Fig. 3 highlights the most changing context words (due to the page limit, we do not show full context) between consecutive turns in a conversation from the CoQA dev. set. As we can see, the hidden representations of context words which are relevant to the consecutive questions are changing most and thus highlighted most. We suspect this is in part because when the focus shifts, the model finds out that the context chunks relevant to the previous turn become less important but those relevant to the current turn become more important. Therefore, the memory updates in these regions are the most active.

5 Conclusion

We proposed a novel Graph Neural Network (GNN) based model, namely GRAPHFLOW, for conversational machine comprehension (MC) which carries over the reasoning output throughout a conversation. Besides, we proposed a simple yet effective graph structure learning technique to dynamically construct a question and conversation history aware context graph at each conversation turn. On three recently released conversational MC benchmarks, our proposed model achieves competitive results compared with previous approaches. Interpretability analysis shows that our model can offer good interpretability for the reasoning process. In the future, we would like to investigate more effective ways of automatically learning graph structures from free text and modeling temporal connections between sequential graphs.

References

[Campos *et al.*, 2019] Jon Ander Campos, Arantxa Otegi, Aitor Soroa, Jan Deriu, Mark Cieliebak, and Eneko Agirre. Conversational qa for faqs. In *NeurIPS*, 2019.

[Chen *et al.*, 2017] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*, 2017.

[Chen *et al.*, 2019a] Yu Chen, Lingfei Wu, and Mohammed J Zaki. Deep iterative and adaptive learning for graph neural networks. *arXiv preprint arXiv:1912.07832*, 2019.

[Chen *et al.*, 2019b] Yu Chen, Lingfei Wu, and Mohammed J Zaki. Reinforcement learning based graph-to-sequence model for natural question generation. *arXiv preprint arXiv:1908.04942*, 2019.

[Chen *et al.*, 2020] Yu Chen, Lingfei Wu, and Mohammed J Zaki. Toward subgraph guided knowledge graph question generation with graph neural networks. *arXiv preprint arXiv:2004.06015*, 2020.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*, pages 1724–1734, 2014.

[Choi *et al.*, 2018] Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. Quac: Question answering in context. *arXiv preprint arXiv:1808.07036*, 2018.

[De Cao *et al.*, 2018] Nicola De Cao, Wilker Aziz, and Ivan Titov. Question answering by reasoning across documents with graph convolutional networks. *arXiv preprint arXiv:1808.09920*, 2018.

[Devlin *et al.*, 2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[Gilmer *et al.*, 2017] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *ICML-Volume 70*, pages 1263–1272. JMLR. org, 2017.

[Hamilton *et al.*, 2017] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, pages 1024–1034, 2017.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Huang *et al.*, 2018] Hsin-Yuan Huang, Eunsol Choi, and Wen-tau Yih. Flowqa: Grasping flow in history for conversational machine comprehension. *arXiv preprint arXiv:1810.06683*, 2018.

[Ju *et al.*, 2019] Ying Ju, Fubang Zhao, Shijie Chen, Bowen Zheng, Xuefeng Yang, and Yunfeng Liu. Technical report on conversational question answering. *arXiv preprint arXiv:1909.10772*, 2019.

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

[Kipf and Welling, 2016] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[Lee *et al.*, 2016] Kenton Lee, Shimi Salant, Tom Kwiatkowski, Ankur Parikh, Dipanjan Das, and Jonathan Berant. Learning recurrent span representations for extractive question answering. *arXiv preprint arXiv:1611.01436*, 2016.

[Li *et al.*, 2015] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[Liu *et al.*, 2018] Pengfei Liu, Shuaichen Chang, Xuanjing Huang, Jian Tang, and Jackie Chi Kit Cheung. Contextualized non-local neural networks for sequence learning. *arXiv preprint arXiv:1811.08600*, 2018.

[Pennington *et al.*, 2014] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543, 2014.

[Qu *et al.*, 2019] Chen Qu, Liu Yang, Minghui Qiu, Yongfeng Zhang, Cen Chen, W Bruce Croft, and Mohit Iyyer. Attentive history selection for conversational question answering. In *CIKM*, pages 1391–1400, 2019.

[Rajpurkar *et al.*, 2016] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[Reddy *et al.*, 2018] Siva Reddy, Danqi Chen, and Christopher D Manning. Coqa: A conversational question answering challenge. *arXiv preprint arXiv:1808.07042*, 2018.

[See *et al.*, 2017] Abigail See, Peter J Liu, and Christopher D Manning. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*, 2017.

[Song *et al.*, 2018] Linfeng Song, Zhiguo Wang, Mo Yu, Yue Zhang, Radu Florian, and Daniel Gildea. Exploring graph-structured passage representation for multi-hop reading comprehension with graph neural networks. *arXiv preprint arXiv:1809.02040*, 2018.

[Sun *et al.*, 2018] Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W Cohen. Open domain question answering using early fusion of knowledge bases and text. *arXiv preprint arXiv:1809.00782*, 2018.

[Wang *et al.*, 2018] Wei Wang, Ming Yan, and Chen Wu. Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. *arXiv preprint arXiv:1811.11934*, 2018.

[Xu *et al.*, 2018a] Kun Xu, Lingfei Wu, Zhiguo Wang, and Vadim Sheinin. Graph2seq: Graph to sequence learning with attention-based neural networks. *arXiv preprint arXiv:1804.00823*, 2018.

[Xu *et al.*, 2018b] Kun Xu, Lingfei Wu, Zhiguo Wang, Mo Yu, Liwei Chen, and Vadim Sheinin. Exploiting rich syntactic information for semantic parsing with graph-to-sequence model. *arXiv preprint arXiv:1808.07624*, 2018.

[Yatskar, 2018] Mark Yatskar. A qualitative comparison of coqa, squad 2.0 and quac. *arXiv preprint arXiv:1809.10735*, 2018.

[Yeh and Chen, 2019] Yi-Ting Yeh and Yun-Nung Chen. Flowdelta: Modeling flow information gain in reasoning for conversational machine comprehension. *arXiv preprint arXiv:1908.05117*, 2019.

[Zhu *et al.*, 2018] Chenguang Zhu, Michael Zeng, and Xuedong Huang. Sdnet: Contextualized attention-based deep network for conversational question answering. *arXiv preprint arXiv:1812.03593*, 2018.