GRAPHFLOW: EXPLOITING CONVERSATION FLOW WITH GRAPH NEURAL NETWORKS FOR CONVERSATIONAL MACHINE COMPREHENSION

Anonymous authors
Paper under double-blind review

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. We propose a novel graph neural network (GNN) based model, namely GRAPHFLOW, which captures conversational flow in the dialog. Specifically, we first propose a new approach to dynamically construct a question-aware context graph from passage text at each turn. We then present a novel flow mechanism to model the temporal dependencies in the sequence of context graphs. The proposed GRAPHFLOW model shows superior performance compared to existing state-of-the-art methods. For instance, GRAPHFLOW outperforms two recently proposed models on the CoQA benchmark dataset: FLOWQA by 2.3% and SDNet by 0.7% on F1 score, respectively. In addition, visualization experiments show that our proposed model can better mimic the human reasoning process for conversational MC compared to existing models.

1 INTRODUCTION

Recent years have observed a surge of interest in conversational machine comprehension (MC). Unlike the setting of MC that requires answering a single question given a passage (aka context), the conversational MC task is to answer the current question in a conversation given a passage and the previous questions and answers. The goal of this task is to mimic real-world situations where humans seek information in a conversational manner.

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, in conversational MC the focus usually shifts as the conversation progresses (Reddy et al., 2018; Choi et al., 2018). Second, many questions refer back to the conversation history via either coreference or ellipsis. Therefore, without fully utilizing the conversation history, one can not understand the current 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 effectively utilize conversation history, including previous questions and/or previous answers. Most existing approaches, however, simply prepend the conversation history to the current question (Reddy et al., 2018; Zhu et al., 2018) or add previous answer locations to the passage (Choi et al., 2018; Yatskar, 2018), and treat the task as a single-turn MC while ignoring the important information from the conversation flow. (Huang et al., 2018) assumed that the hidden representations generated during the previous reasoning processes potentially capture important information for answering the previous questions, and thus provide additional clues for answering the current question. They proposed an *Integration-Flow* (IF) mechanism to first process sequentially in passage, in parallel of question turns and then process sequentially in question turns, in parallel of passage words. Their FLOWQA model achieves strong empirical results on two benchmarks (i.e., CoQA and QuAC) (Reddy et al., 2018; Choi et al., 2018).

However, the IF mechanism is not quite natural since it does not mimic how humans perform reasoning. This is because when humans execute such task, they typically do not first perform reasoning in parallel for each question, and then refine the reasoning results across different turns. This may partially explain why this strategy is inefficient because the results of previous reasoning processes are not incorporated into the current reasoning process. As a result, the reasoning performance at each turn is only slightly improved by the hidden states of the previous reasoning process, even though they use stacked IF layers to try to address this problem.

To address the aforementioned issues, we propose GRAPHFLOW, a Graph Neural Network (GNN) based model for conversational MC. As shown in Fig. 1, GRAPHFLOW consists of three components, *Encoding Layer*, *Reasoning Layer*, and *Prediction Layer*. The *Encoding Layer* encodes conversation history and the context text that aligns question embeddings. The *Reasoning Layer* dynamically constructs a question-aware context graph at each turn, and then applies GNNs to process the sequence of context graphs. In particular, 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, which is closer to how humans perform reasoning in a conversational setting, compared to existing approaches. The *Prediction Layer* predicts the answers based on the matching score of the question embedding and the learned graph node embeddings for context at each turn.

We highlight our contributions as follows:

- We propose a novel graph neural network based model, namely GRAPHFLOW, for conversational MC which captures conversational flow in the dialog.
- We dynamically construct a context graph that consists of each passage word as a node, which encodes not only the passage itself, but also the question as well as the conversation history. Besides, to our best knowledge, we are the first to treat context as a graph of words in the literature of MC.
- We propose a novel flow mechanism to process a sequence of context graphs, which models the temporal dependencies between consecutive context graphs.
- On two public benchmarks (i.e., CoQA and QuAC), the proposed model shows superior performance compared to existing state-of-the-art methods. For instance, GRAPHFLOW outperforms two recently proposed models FLOWQA by 2.3% F1 and SDNet by 0.7% F1 on benchmark dataset CoQA, respectively.
- Our interpretability analysis shows that our model can better mimic human reasoning process on this task.

2 THE GRAPHFLOW APPROACH

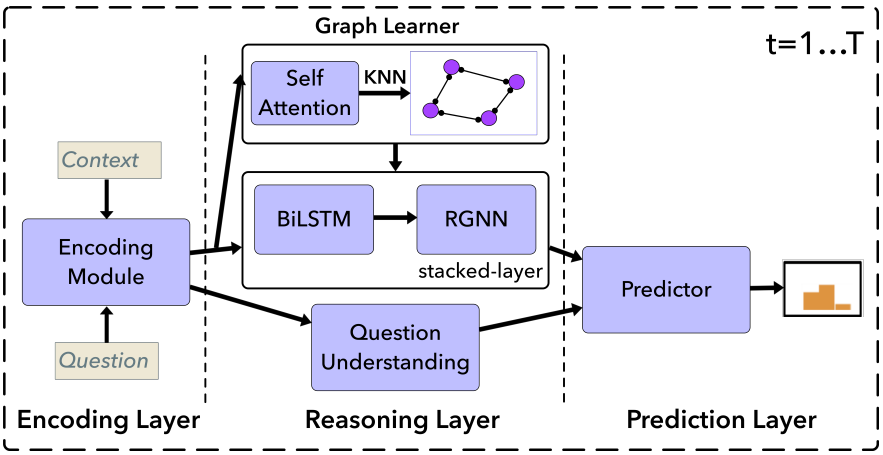

Figure 1: Overall architecture of the proposed model. Best viewed in color.

2.1 ENCODING LAYER

Let us denote C as the context which consists of a sequence of words $\{c_1, c_2, ..., c_m\}$ and Q_i as the question at the i-th turn which consists of a sequence of words $\{q_1^{(i)}, q_2^{(i)}, ..., q_n^{(i)}\}$. We apply an effective encoding layer to encode the context and the question, which additionally exploits the interactions (i.e., soft alignment) between them as well as conversation history (i.e., previous question-answer pairs). The details of the encoder are given next.

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

Pretrained word embeddings We use 300-dim GloVe (Pennington et al., 2014) embeddings as well as 1024-dim BERT (Devlin et al., 2018) embeddings to embed each word in the context and the question. Compared to GloVe embeddings, BERT embeddings better utilize the contextual information and can learn different embeddings for the same word type within different context.

Aligned question embeddings Exact matching only matches words on the surface form, we further apply an attention mechanism to learn the soft alignment between context words and question words. Following (Lee et al., 2016) and recent work, for each context word c_j at the i-th turn, we incorporate an aligned question embedding $f_{align}(c_j^{(i)}) = \sum_k a_{j,k}^{(i)} \mathbf{g}_k^{Q_i}$ where $\mathbf{g}_k^{Q_i}$ is the GloVe embedding of question word $q_k^{(i)}$ and $a_{j,k}^{(i)}$ is an attention score between context word c_j and question word $q_k^{(i)}$. Here we define the attention score $a_{j,k}^{(i)}$ as,

$$a_{j,k}^{(i)} \propto \exp(\text{ReLU}(\mathbf{W}\mathbf{g}_j^C)^T \text{ReLU}(\mathbf{W}\mathbf{g}_k^{Q_i})) \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{d \times 300}$ is a trainable model parameter, d is 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 $\text{Align}(\mathbf{A}, \mathbf{B}, \mathbf{C})$, meaning that an attention matrix is computed between two sets of vectors $\mathbf{A}$ and $\mathbf{B}$, which is later used to get a linear combination of vector set $\mathbf{C}$. Hence we can reformulate the above alignment as $f_{\text{align}}(C^{(i)}) = \text{Align}(\mathbf{g}^C, \mathbf{g}^{Q_i}, \mathbf{g}^{Q_i})$.

Conversation history Conversation history is crucial for understanding the current question in this task. (Choi et al., 2018) found it useful to concatenate a feature vector $f_{\text{ans}}(c_j^{(i)})$ encoding previous N answer locations to the context word embeddings. We adopt this strategy, and additionally prepend previous N question-answer pairs to the current question. When prepending conversation history, a common choice is to separate the current question from conversation history using certain special token, which does not work well in practice as observed by (Choi et al., 2018). We find a more effective strategy to do the separation which is to concatenate a turn marker embedding $f_{\text{turn}}(q_k^{(i)})$ to each word vector in the augmented question. The turn marker is intended to indicate which turn the word belongs to (e.g., i indicates the previous i-th turn).

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 $f_{\text{ling}}(c_j^{(i)})$, GloVe vector $\mathbf{g}_j^C$, BERT vector 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 $\mathbf{w}_k^{Q_i}$ which is a concatenation of GloVe vector $\mathbf{g}_k^{Q_i}$, BERT vector $\text{BERT}_k^{Q_i}$ 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 $\mathbf{w}_k^{Q_i}$, respectively at the i-th turn.

2.2 REASONING LAYER

When performing reasoning over context, unlike most previous methods that solely regard context as a sequence of words, we opt to treat context as a "graph" of words, and hence it is straightforward to apply GNNs to process the context graph. Our Reasoning Layer consists of the *Question Understanding* module, *Context Graph Learning* module and *Context Graph Reasoning* module.

2.2.1 QUESTION UNDERSTANDING

For each question Q_i, we apply a BiLSTM (Hochreiter & Schmidhuber, 1997) to the raw question embeddings $\mathbf{W}^{Q_i}$ to obtain contextualized embeddings $\mathbf{Q}_i \in \mathbb{R}^{d \times n}$.

$$\mathbf{Q}_i = \mathbf{q}_1^{(i)}, ..., \mathbf{q}_n^{(i)} = \text{BiLSTM}(\mathbf{W}^{Q_i}) \tag{2}$$

Each question is then represented as a weighted sum of word vectors in the question via a simple self attention mechanism.

$$\widetilde{\mathbf{q}}^{(i)} = \sum_k a_k^{(i)} \mathbf{q}_k^{(i)}, \text{ where } a_k^{(i)} \propto \exp(\mathbf{w}^T \mathbf{q}_k^{(i)}) \tag{3}$$

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

Finally, to capture the dependency among questions at different turns, we encode question history sequentially in question turns with a LSTM to generate history-aware question vectors.

$$\mathbf{p}_1, ..., \mathbf{p}_T = \text{LSTM}(\widetilde{\mathbf{q}}^{(1)}, ..., \widetilde{\mathbf{q}}^{(T)}) \tag{4}$$

The output hidden states of the LSTM network $\mathbf{p}_1, ..., \mathbf{p}_T$ will be used for predicting answers.

2.2.2 CONTEXT GRAPH LEARNING

The intrinsic context graph structure is unfortunately unknown in the MC task. Moreover, intuitively speaking, the context graph structure might vary across different turns by considering the change of questions and conversation history. Most existing applications of GNNs take as input ground-truth graphs or manually constructed graphs which have some limitations. First, the ground-truth graphs are not always available. Second, the error in the manual construction process can be propagated to the subsequent GNN-based models and cannot be reduced during the learning process. In this work, we choose to automatically construct graphs from raw context, which will be combined with the remaining of the system to make the whole learning system end-to-end trainable. The difference between manual graph construction and automatic graph construction is analogous to the difference between the feature engineering + Machine Learning paradigm and the Deep Learning end-to-end paradigm.

We dynamically build a weighted graph to model semantic relationships among context words at each turn in a conversation. We make the process of building such a context graph depend on not only the semantic meanings of context words, but also on the question being asked, as well as the conversation history, so as to help better answer the question.

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 $\mathbf{A}_C^{(i)}$, serving as a weighted adjacency matrix for the context graph, defined as,

$$\mathbf{A}_C^{(i)} = \text{ReLU}(\mathbf{U}\mathbf{W}_C^{(i)})^T \text{ReLU}(\mathbf{U}\mathbf{W}_C^{(i)}) \tag{5}$$

where $\mathbf{U}$ is a $d \times d_c$ trainable weight and d_c is the embedding size of $\mathbf{w}_{c_j}^{(i)}$.

Considering that a fully connected context graph is not only computationally expensive but also makes little sense for reasoning, a simple KNN-style graph sparsification operation is conducted to extract a sparse and weighted graph from the fully connected one. To be concrete, given a learned attention matrix $\mathbf{A}_C^{(i)}$, 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, and apply a softmax function to these selected adjacency matrix elements to get a sparse and normalized adjacency matrix $\widetilde{\mathbf{A}}_C^{(i)}$.

$$\widetilde{\mathbf{A}}_C^{(i)} = \text{softmax}(\text{topk}(\mathbf{A}_C^{(i)})) \tag{6}$$

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.

2.2.3 CONTEXT GRAPH REASONING

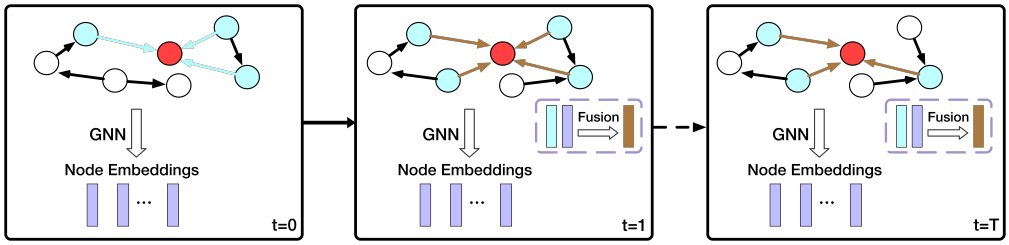

Figure 2: Architecture of the proposed Recurrent Graph Neural Network (RGNN) for processing a sequence of context graphs. Best viewed in color.

Given the context graphs constructed at each turn, we propose a novel *Recurrent Graph Neural Network* (RGNN) to sequentially process a sequence of graphs, as shown in Fig. 2. Conceptually, 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. As we advance in a sequence of graphs, we process each graph using a shared GNN cell and the output of the GNN cell will be used when processing the next 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 RGNNs are as follows. At the i-th turn, before we apply a GNN to the context graph $\mathcal{G}_i$, we update node embeddings by fusing both the original node information $\mathbf{C}_i^{l-1}$ and the updated node information at the previous turn $\mathbf{C}_{i-1}^{l}$ via a fusion function.

$$\mathbf{C}_i^l = \text{GNN}(\bar{\mathbf{C}}_i^{l-1}, \tilde{\mathbf{A}}_C^{(i)}) \quad \bar{\mathbf{C}}_{i,j}^{l-1} = \text{Fuse}(\mathbf{C}_{i,j}^{l-1}, \mathbf{C}_{i-1,j}^{l}) \tag{7}$$

where l is the RGNN layer index. Note that we can stack multiple RGNN layers to enhance the performance if necessary. The fusion function is designed as a gated sum of two information sources,

$$\text{Fuse}(\mathbf{a}, \mathbf{b}) = \mathbf{z} * \mathbf{a} + (1 - \mathbf{z}) * \mathbf{b} \quad \mathbf{z} = \sigma(\mathbf{W}_z[\mathbf{a}; \mathbf{b}; \mathbf{a} * \mathbf{b}; \mathbf{a} - \mathbf{b}] + \mathbf{b}_z) \tag{8}$$

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. Note that we set $\bar{\mathbf{C}}_0^{l-1} = \mathbf{C}_0^{l-1}$ as we will not incorporate any historical information at the first turn. Notably, even though there is no direct link between c_j at the $(i-1)$-th turn and c_k at the i-th turn, the information flow can be propagated between them as the GNN progresses, which means information can be exchanged among all the objects spatially and temporally.

We use Gated Graph Neural Networks (GGNN) (Li et al., 2015) as our GNN module, but the framework is agnostic to the particular choice of GNN module. In GGNN we can do multi-hop message passing through the graph to capture long-range dependency where the same set of network parameters are shared at every hop of computation. At each hop of GGNN computation, for every node in the graph, we apply an aggregation function which takes as input a set of neighboring node embeddings and outputs an aggregation vector. In our framework, we compute a weighted average for aggregation where the weights come from the normalized adjacency matrices $\tilde{\mathbf{A}}_C^{(i)}$. Then, a Gated Recurrent Unit (GRU) (Cho et al., 2014) is used to update the node embeddings by incorporating the aggregation information. 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 $\mathbf{C}^l = \text{RGNN}(\mathbf{C}^{l-1}, \widetilde{\mathbf{A}}_C)$ which takes as input a sequence of graph node embeddings $\mathbf{C}^{l-1}$ as well as a sequence of the normalized adjacency matrices $\widetilde{\mathbf{A}}_C$, and outputs a sequence of updated graph node embeddings $\mathbf{C}^l$.

While a GNN is responsible for modeling the global interactions among context words, modeling local interactions among 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 the context words, that is, $\mathbf{C}_i^0 = \text{BiLSTM}(\mathbf{W}_C^{(i)})$, and we then use the output $\mathbf{C}_i^0$ 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 one RGNN layer on low level representations of the context and another RGNN layer on high level representations of the context, as formulated in the following.

$$\begin{aligned}
\mathbf{C}^1 &= \text{RGNN}(\mathbf{C}^0, \widetilde{\mathbf{A}}_C) \\
\mathbf{H}_i^C &= [\mathbf{C}_i^1; \mathbf{g}^C; \text{BERT}^C] \\
\mathbf{H}_i^Q &= [\mathbf{Q}_i; \mathbf{g}^{Q_i}; \text{BERT}^{Q_i}] \\
f_{\text{align}}^2(C^{(i)}) &= \text{Align}(\mathbf{H}_i^C, \mathbf{H}_i^Q, \mathbf{Q}_i)] \\
\widetilde{\mathbf{C}}_i^1 &= \text{BiLSTM}([\mathbf{C}_i^1; f_{\text{align}}^2(C^{(i)})]) \\
\mathbf{C}^2 &= \text{RGNN}(\widetilde{\mathbf{C}}^1, \widetilde{\mathbf{A}}_C)
\end{aligned} \tag{9}$$

2.3 Prediction Layer

We predict answer spans by computing the start and end probabilities $P_{i,j}^S$ and $P_{i,j}^E$ of the j-th context word for the i-th question. To predict the start index, the probability is calculated by,

$$P_{i,j}^S \propto \exp({\mathbf{c}_{i,j}^2}^T \mathbf{W}_S \mathbf{p}_i) \tag{10}$$

where $\mathbf{W}_S$ is a $d \times d$ trainable weight. Next, $\mathbf{p}_i$ is passed to a GRU by incorporating context summary and converted to $\widetilde{\mathbf{p}}_i$.

$$\widetilde{\mathbf{p}}_i = \text{GRU}(\mathbf{p}_i, \sum_j P_{i,j}^S \mathbf{c}_{i,j}^2) \tag{11}$$

Then, the end probability is calculated by,

$$P_{i,j}^E \propto \exp({\mathbf{c}_{i,j}^2}^T \mathbf{W}_E \widetilde{\mathbf{p}}_i) \tag{12}$$

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,

$$\widetilde{\mathbf{C}}_i^2 = [f_{\text{mean}}(\mathbf{C}_i^2); f_{\text{max}}(\mathbf{C}_i^2)] \quad P_i^C = \sigma(f_c(\mathbf{p}_i)(\widetilde{\mathbf{C}}_i^2)^T) \tag{13}$$

where f_c is a dense layer which maps a d-dim vector to a $(\text{num\_class} \times 2d)$-dim vector. Further, σ is a sigmoid function for binary classification and a softmax function for multi-class classification. We use $\widetilde{\mathbf{C}}_i^2$ to represent the whole context at the i-th turn which is a concatenation of average pooling and max pooling outputs of $\mathbf{C}_i^2$.

2.4 Training and Inference

The training objective is defined as the cross entropy loss of both text span prediction (if the question requires it) and answer type prediction.

$$\mathcal{L} = -\sum_i I_i^S (log(P_{i,s_i}^S) + log(P_{i,e_i}^E)) + \log P_{i,t_i}^C \tag{14}$$

where I_i^S indicates whether this question requires text span prediction, s_i and e_i are the ground-truth start and end positions of the answer span, and t_i indicates the ground-truth answer type.

During inference, we first use P_i^C to predict whether the question requires text span prediction. If yes, we predict the span to be s_i, e_i with maximum $P_{i,s_i}^S P_{i,e_i}^E$ subject to the length constraint on the span.

3 EXPERIMENTS

In this section, we conduct an extensive evaluation of our proposed model against state-of-the-art conversational MC models. We use two popular benchmarks, described below. The implementation of the model will be publicly available soon. Detailed description of model settings is provided in Appendix A.

Table 1: Model and human performance (% in F1 score) on the CoQA test set.

	Child.	Liter.	Mid-High.	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
SDNet	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 2: Model and human performance (in %) 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	**64.9**	**60.3**	5.1
Human	80.8	100	100

3.1 DATA AND METRICS

Although CoQA is released as an abstractive CMC dataset, Yatskar (2018) shows that the extractive approach is also effective for CoQA. Thus, we also use our extractive approach on CoQA. To handle answer types in CoQA, we predict the probability distribution of the answer type (SPAN, YES, NO, and UNANSWERABLE) and replace the predicted span with yes, no, or unknown tokens ex- cept for the SPAN answer type. In QuAC, the unanswerable questions are handled as an answer span (P contains a special token), and the type prediction for yes/no questions is not evaluated on the leaderboard. Therefore, we skip the answer type prediction step.

CoQA CoQA contains 127k questions with answers, obtained from 8k conversations. In CoQA, answers are in free-form and hence are not necessarily text spans from the context (i.e., 33.2% of the questions have abstractive answers). Although this calls for a generation approach, following previous work (Yatskar, 2018; Huang et al., 2018; Zhu et al., 2018), as detailed in Section 2.3, we adopt an extractive approach with additional answer type classifiers to handle non-extractive questions. The average length of questions is only 5.5 words (i.e., 70% questions have coreference or ellipsis), which means conversation history is important for better understanding those questions. The average number of turns per dialog is 15.2. Notably, in the testing set, there are two out-of-domain datasets which are reserved for testing only.

QuAC QuAC contains 98k questions with answers, obtained from 13k conversations. Unlike CoQA, all the answers in QuAC are text spans from the context. The average length of questions is 6.5 and there are on average 7.2 questions per dialog. The average length of QuAC context is 401 which is longer than that of CoQA which is 271. The average length of QuAC answers is 14.6 which is also longer than that of CoQA which is 2.7.

The main evaluation metric is F1 score which is the harmonic mean of precision and recall at word level between the predicted answer and ground truth. In addition, for QuAC 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, respectively. Please refer to (Reddy et al., 2018; Choi et al., 2018) for details of these metrics.

3.2 MODEL COMPARISON

Baseline methods We compare our approach with the following baseline methods: PGNet (See et al., 2017), DrQA (Chen et al., 2017), DrQA+PGNet (Reddy et al., 2018), BiDAF++ (Yatskar, 2018), FLOWQA (Huang et al., 2018) and SDNet (Zhu et al., 2018). We will discuss the details of these methods in Section 4.1.

Results As shown in Section 3 and Section 3, our model consistently outperforms these state-of-the-art baselines in terms of F1 score. In particular, GRAPHFLOW yields improvement over all existing models on both datasets by at least +0.7% F1 on CoQA and +0.8% F1 on QuAC, respectively. Compared with FLOWQA which is also based on the flow idea, our model improves F1 by 2.3% on CoQA and 0.8% on QuAC, which demonstrates the superiority of our *RGNN* mechanism over the *Integration-Flow* mechanism. Compared with SDNet which relies on sophisticated inter-attention and self-attention mechanisms, our model improves F1 by 0.7% on CoQA[1].

3.3 ABLATION STUDY

Table 3: Ablation study: model performance (in %) on the CoQA dev. 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
GRAPHFLOW (1-His)	78.2
GRAPHFLOW (0-His)	76.7

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 . When he got closer he also saw that there were five white chickens sitting on the fence . The fence was painted blue and had some dirty black spots on it . Billy wondered where the dirty spots had come . Soon he got close to the chickens and they got scared . All five chickens flew away and went to eat some food . 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 it was a good deal so he went home and cooked his brother dinner . His brother was very happy with the dinner . Billy 's mom was also very happy .

Q2: Why? -> Q3: For what?

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 . When he got closer he also saw that there were five white chickens sitting on the fence . The fence was painted blue and had some dirty black spots on it . Billy wondered where the dirty spots had come . Soon he got close to the chickens and they got scared . All five chickens flew away and went to eat some food . 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 it was a good deal so he went home and cooked his brother dinner . His brother was very happy with the dinner . Billy 's mom was also very happy .

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

[1]They did not report the results on QuAC.

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, – 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).

Section 3.3 shows the contributions of the above components on the CoQA development set. We find that the pretrained BERT embedding has the most impact on the performance, which again demonstrates the power of large-scale pretrained language models. Our proposed RGNN module also 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. We also notice that explicitly adding conversation history to the current turn helps the model performance by comparing GRAPHFLOW (2-His), GRAPHFLOW (1-His) and GRAPHFLOW (0-His). 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. We conjecture this is partially because the turn transitions in a conversation are usually smooth and marking the previous answer locations helps the model better identify relevant context chunks for the current question.

3.4 INTERPRETABILITY ANALYSIS

Here we visualize the memory bank (i.e., an m by d matrix) which stores the hidden representations (and thus reasoning output) of the context throughout a conversation. While directly visualizing the hidden representations is difficult, thanks to the flow-based mechanism introduced into our model, we instead visualize the changes of hidden representations of context words between consecutive turns. We expect that the most changing parts of the context should be those which are relevant to the questions being asked and therefore should probably be able to indicate shifts of the focus in a conversation.

Following (Huang et al., 2018), we visualize this by computing the cosine similarity of the 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)[2]. Fig. 3 highlights the most changing context words 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 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. Obviously, this makes the model easier to answer follow-up questions. As we observe in our visualization experiments, in conversations extensively involving coreference or ellipsis, our model can still perform reasonably well. We refer the interested readers to the supplementary material for complete visualization examples.

4 RELATED WORK

4.1 CONVERSATIONAL MC

Many methods have been proposed to utilize conversation history in the literature of conversational MC. (Reddy et al., 2018; Zhu et al., 2018) concatenate previous questions and answers to the current question. (Choi et al., 2018) concatenate 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, as claimed by (Huang et al., 2018), these methods ignore previous reasoning processes performed by the model when reasoning at the current turn. Instead, they propose

[2]For better visualization, we apply an attention threshold of 0.3 to highlight only the dramatically changing context words.

the idea of *Integration-Flow* to allow rich information in the reasoning process to flow through a conversation.

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 RC 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.

4.2 Graph Neural Networks

Over the past few years, graph neural networks (GNNs) (Kipf & Welling, 2016; Gilmer et al., 2017; Hamilton et al., 2017; Li et al., 2015; Xu et al., 2018a) have drawn increasing attention since they extend traditional Deep Learning approaches to non-euclidean data such as graph-structured data. Recently, GNNs have been applied to various question answering tasks including visual question answering (VQA) (Teney et al., 2017; Norcliffe-Brown et al., 2018), knowledge base question answering (KBQA) (Sun et al., 2018) and MC (De Cao et al., 2018; Song et al., 2018; Xu et al., 2018c), and have shown advantages over traditional approaches. For tasks where the graph structure is unknown, (De Cao et al., 2018; Song et al., 2018; Xu et al., 2018b) construct a static graph where entity mentions in a passage are nodes of this graph and edge information is determined by coreferences of entity mentions. (Norcliffe-Brown et al., 2018) dynamically construct a graph which contains all the visual objects appearing in an image. In parallel to our work, (Liu et al., 2018) also dynamically construct a graph of all words from free text.

5 Conclusion

We proposed a novel GNNs-based model, namely GraphFlow, for conversational MC which carries over the reasoning output throughout a conversation. On two recently released conversational MC benchmarks, our proposed model achieves superior results over previous approaches. Interpretability analysis shows that the model can better mimic the human reasoning process for conversational MC compared to existing models.

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

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

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*, pp. 1724–1734, 2014.

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.

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

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1263–1272. JMLR. org, 2017.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pp. 1024–1034, 2017.

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

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

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

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

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

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.

Will Norcliffe-Brown, Stathis Vafeias, and Sarah Parisot. Learning conditioned graph structures for interpretable visual question answering. In *Advances in Neural Information Processing Systems*, pp. 8344–8353, 2018.

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

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

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

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

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.

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.

Damien Teney, Lingqiao Liu, and Anton van den Hengel. Graph-structured representations for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2017.

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.

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

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*, 2018b.

Kun Xu, Lingfei Wu, Zhiguo Wang, Mo Yu, Liwei Chen, and Vadim Sheinin. Sql-to-text generation with graph-to-sequence model. *arXiv preprint arXiv:1809.05255*, 2018c.

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

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

A MODEL SETTINGS

We construct the vocabulary of words from the training set but filter out those infrequent words (i.e., word count less than 5) to reduce the vocabulary size. The embedding sizes of POS, NER, exact matching and turn marker embeddings are set to 12, 8, 3 and 3, respectively. We fix the pretrained GloVe vectors since which we find gives better results than the fine-tuning strategy. 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 3 for QuAC. During training, we apply dropout after the embedding layers (0.3 for GloVe and 0.4 for BERT). A dropout rate of 0.3 is also applied after the output of all RNN layers. We use Adamax (Kingma & Ba, 2014) as the optimizer and the learning rate is set to 0.001. We reduce the learning rate by a factor of 0.5 if the validation F1 score has stopped improving every one epoch. We stop the training when no improvement is seen for 10 consecutive epochs. We clip the gradient at length 10. 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. When doing text span prediction, the span is constrained to have a maximum length of 12 for CoQA and 35 for QuAC. All these hyper-parameters are tuned on the development set.