FLOWQA: GRASPING FLOW IN HISTORY FOR CONVERSATIONAL MACHINE COMPREHENSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Conversational machine comprehension requires deep understanding of the conversation history. To enable traditional, single-turn models to encode the history comprehensively, we introduce FLOW, a mechanism that can incorporate intermediate representations generated during the process of answering previous questions, through an alternating parallel processing structure. Compared to shallow approaches of concatenating previous questions/answers as input, FLOW integrates the latent semantics of the conversation history more deeply, and effectively stacks single-turn models along the conversational flow. Our model, FLOWQA, shows superior performance on two recently proposed conversational challenges (+7.2% F_1 on CoQA and +4.0% on QuAC). The importance of encoding history also shows in other tasks. By reducing sequential instruction understanding to conversational machine comprehension, FLOWQA outperforms the best models on all three domains in SCONE, with +1.8% to +4.4% improvement in accuracy.

1 INTRODUCTION

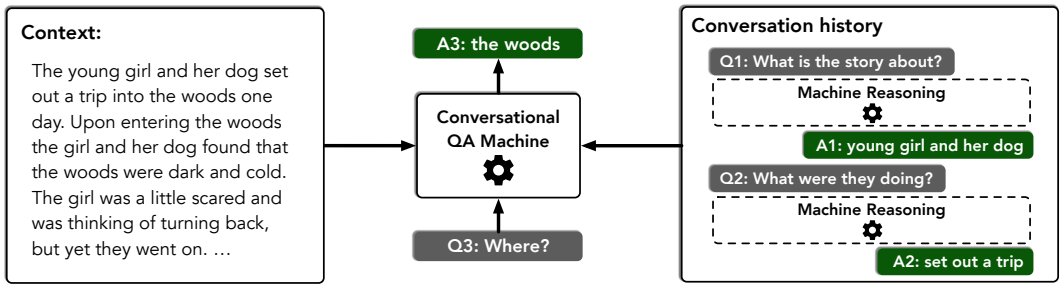

Figure 1: An illustration of conversational machine comprehension with an example from the Conversational Question Answering Challenge dataset (CoQA).

Humans seek information in a *conversational* manner, by asking follow-up questions for additional information based on what they have already learned. Recently proposed conversational machine comprehension (MC) datasets (Reddy et al., 2018; Choi et al., 2018) aim to enable models to assist in such information seeking dialogs. They consist of a sequence of question/answer pairs where questions can only be understood along with the conversation history. Figure 1 illustrates this new challenge. Existing approaches take a single-turn MC model and augment the current question and context with the previous questions and answers (Choi et al., 2018; Reddy et al., 2018).[1] However, this offers only a partial solution, ignoring previous *reasoning*[2] processes performed by the model.

We present FLOWQA, a model designed for conversational machine comprehension. FLOWQA consists of two main components: a base neural model for single-turn MC and a FLOW mechanism that *encodes the conversation history*. Instead of just using the shallow history like previous questions and answers, we feed the model with the entire hidden representations generated during the process of answering *previous* questions. These hidden representations potentially capture related information, such as phrases and facts in the context, for answering the previous questions, and

[1] Detailed explanation of existing models is in Section 4.

[2] We use "reasoning" to refer to the model process of finding the answer.

hence provide additional clues on what the current conversation is revolving around. This FLOW mechanism is also remarkably effective at tracking the world states for sequential instruction understanding (Long et al., 2016): after mapping world states as context and instructions as questions, FLOWQA can interpret a sequence of inter-connected instructions and generate corresponding world state changes as answers. The FLOW mechanism can be viewed as stacking single-turn QA models along the dialog progression (i.e., the question turns) and building information flow along the dialog. This information transfer happens for *each context word*, allowing rich information in the reasoning process to flow. This design is analogous to recurrent neural networks, where each single update unit is now an entire question answering process. Because there are two recurrent structures in our modeling, one in the context for each question and the other in the conversation progression, a naive implementation leads to a highly unparallelizable structure. To handle this issue, we propose an alternating parallel processing structure, which alternates between sequentially processing one dimension in parallel of the other dimension, and thus speeds up training significantly.

FLOWQA achieves strong empirical results on conversational machine comprehension tasks, and improves the state of the art on various datasets (from 67.8% to 75.0% on CoQA and 60.1% to 64.1% on QuAC). Perhaps more impressively, although designed for conversational machine comprehension, FLOWQA also shows superior performance on a seemingly different task – understanding a sequence of natural language instructions (framed previously as a sequential semantic parsing problem). When tested on SCONE (Long et al., 2016), FLOWQA outperforms all existing systems in three different domains, resulting in a range of accuracy improvement from +1.8% to +4.4%.

2 BACKGROUND: MACHINE COMPREHENSION TASKS AND MODELS

In this section, we introduce the task formulations of machine comprehension in both single-turn and conversational settings, and discuss the main ideas of state-of-the-art MC models.

2.1 TASK FORMULATION

Given an evidence document (context) and a question, the task is to find the answer. The context $\boldsymbol{C} = \{c_1, c_2, \dots c_m\}$ is described as a sequence of m words and the question $\boldsymbol{Q} = \{q_1, q_2 \dots q_n\}$ a sequence of n words. In the extractive setting, the answer $\boldsymbol{A}$ must be a span in the context. Conversational machine comprehension is a generalization of the single-turn setting: the agent needs to answer multiple, potentially inter-dependent questions in an interactive fashion. The meaning of the current question may depend on the conversation history (e.g., in Fig. 1, a question such as 'Where?' cannot be answered in isolation). Thus, previous conversational history (i.e., question/answer pairs) is provided in addition to the context and the current question.

2.2 BASIC DESIGN OF MACHINE COMPREHENSION MODELS

For single-turn MC, many top-performing models share a similar architecture, consisting of four major components: (1) question encoding, (2) context encoding, (3) reasoning, and finally (4) answer prediction. Initially the word embeddings (e.g., Pennington et al., 2014; Peters et al., 2018) of question tokens $\boldsymbol{Q}$ and context tokens $\boldsymbol{C}$ are taken as input and fed into contextual integration layers, such as LSTMs (Hochreiter & Schmidhuber, 1997) or self attentions (Yu et al., 2018), to *encode* the question and context. Multiple integration layers provide contextualized representations of context, and are often inter-weaved with attention, which inject question information. The context integration layers thus produce a series of query-aware hidden vectors for each word in the context. Together, the context integration layers can be viewed as conducting implicit *reasoning* to find the answer candidates. The final output is fed into the *answer prediction* layer to select the answer. To adapt to the conversational setting, existing methods incorporate previous question/answer pairs into the current question and context encoding without modifying higher-level (reasoning and answer prediction) layers of the model.

3 FLOWQA

Our model aims to incorporate the conversation history more comprehensively via a conceptually simple FLOW mechanism. We first introduce the concept of FLOW (Section 3.1), propose the

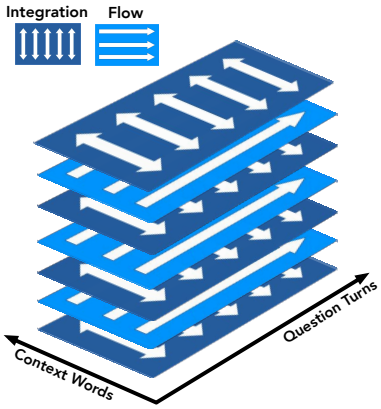

Figure 2: Alternating computational structure between context integration (RNN over context) and FLOW (RNN over question turns).

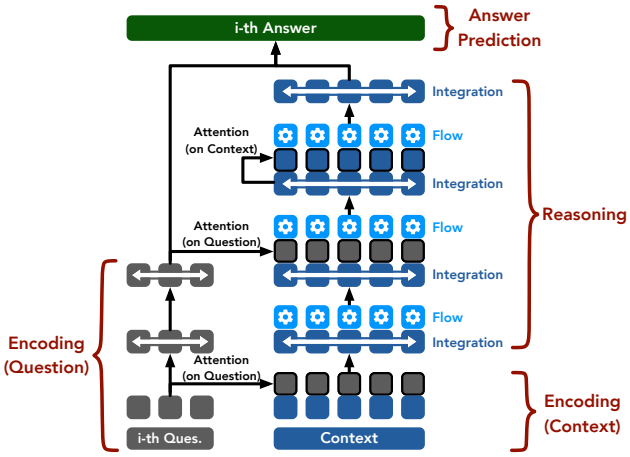

Figure 3: An illustration of the architecture for FLOWQA.

INTEGRATION-FLOW layers (Section 3.2), and present an end-to-end architecture for conversational machine comprehension, FLOWQA (Section 3.3).

3.1 CONCEPT OF FLOW

Successful conversational MC models should grasp the conversation flow, such as the main topics revolving the dialog, relevant events being discussed, or related facts about the topic. Since the conversation is based on the context $\boldsymbol{C}$, we consider the conversation flow as a representation based of the context tokens. Conversation flow can be inferred by just using previous question answer pairs, but for machine to better grasp the conversation flow, we utilize the intermediate machine process for answering previous questions (See Fig. 1). In MC models, the intermediate machine process is captured in the several context *integration* layers (often BiLSTMs), which locate the answer candidates in the context. Our model considers this intermediate representations, $\boldsymbol{C}_i^h$, generated during the h-th context integration layer of the reasoning component for the i-th question. FLOW builds information flow from the intermediate representation $\boldsymbol{C}_1^h, \ldots, \boldsymbol{C}_{i-1}^h$ generated for the previous question $\boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_{i-1}$ to the current process for answering $\boldsymbol{Q}_i$, for every h and i.

3.2 INTEGRATION-FLOW LAYER

A naive sequential implementation of FLOW would pass the output hidden vectors from each integration layer during the $(i-1)$-th question turn to the corresponding integration layer for $\boldsymbol{Q}_i$. This is highly unparalleled, as the contexts have to be read in order, and the question turns have to be processed sequentially. To achieve better parallelism, we alternate between them: **context integration**, processing sequentially in context, in parallel of question turns; and **flow**, processing sequentially in question turns, in parallel of context words (see Fig. 2). This gives a 5 to 10 times speedup[3] at training time. Throughout this section, we will use i $(1 \leq i \leq t)$ to iterate over question, j $(1 \leq j \leq m)$ to iterate over context, where t is the length of the dialog and m is the length of the context. Below we describe the implementation of an INTEGRATION-FLOW (IF) layer, which is composed of a context integration layer and a FLOW component.

Context Integration We pass the current context representation $\boldsymbol{C}_i^h$ for each question i into a BiLSTM layer. All question i $(1 \leq i \leq t)$ are processed in parallel during training.

$$\hat{\boldsymbol{C}}_i^h = \hat{c}_{i,1}^h, \ldots, \hat{c}_{i,m}^h = \text{BiLSTM}([\boldsymbol{C}_i^h]) \tag{1}$$

FLOW After the integration, we have t context sequences of length m, one for each question. We reshape it to become m sequences of length t, one for each context word. We then pass each sequence into a GRU[4] so the entire intermediate representation for answering the previous questions

[3]The speedup depends on the average length of the conversations.

[4]We use GRU because it is faster and performs comparably to LSTM based on our preliminary experiments.

can be used when processing the current question. We only consider the forward direction since we do not know the $(i+1)$-th question when answering the i-th question. All context word j $(1 \leq j \leq m)$ are processed in parallel.

$$f_{1,j}^{h+1}, \ldots, f_{t,j}^{h+1} = \mathrm{GRU}(\hat{c}_{1,j}^{h}, \ldots, \hat{c}_{t,j}^{h}) \tag{2}$$

We reshape the outputs from the FLOW layer back, and concatenate them to the output of the integration layer.

$$F_i^{h+1} = \{f_{i,1}^{h+1}, \ldots, f_{i,m}^{h+1}\} \tag{3}$$

$$\boldsymbol{C}_i^{h+1} = c_{i,1}^{h+1}, \ldots, c_{i,m}^{h+1} = [\hat{c}_{i,1}^{h}; f_{i,1}^{h+1}], \ldots, [\hat{c}_{i,m}^{h}; f_{i,m}^{h+1}] \tag{4}$$

In summary, this process takes $\boldsymbol{C}_i^h$ and generates $\boldsymbol{C}_i^{h+1}$, which will be used for further contextualization to predict the start and end answer span tokens. When FLOW is removed, the IF layer becomes a regular context integration layer and in this case, a single layer of BiLSTM.

3.3 DETAILED DESIGN OF FLOWQA

We construct our conversation MC model, FLOWQA, based on the single-turn MC structure (Sec. 2.2) with fully-aware attention (Huang et al., 2018). The full architecture is shown in Fig. 3. In this section, we describe its main components: initial encoding, reasoning and answer prediction.

3.3.1 QUESTION/CONTEXT ENCODING

Word Embedding We embed the context into a sequence of vectors, $\boldsymbol{C} = \{c_1, \ldots, c_m\}$ with pretrained GloVe (Pennington et al., 2014), CoVE (McCann et al., 2017) and ELMo (Peters et al., 2018) embeddings. Similarly, each question at the i-th turn is embedded into a sequence of vectors $\boldsymbol{Q}_i = \{q_{i,1}, \ldots, q_{i,n}\}$, where n is the maximum question length for all questions in the conversation.

Attention (on Question) Following DrQA (Chen et al., 2017), for each question, we compute attention in the word level to enhance context word embeddings with question.

$$g_{i,j} = \sum\nolimits_k \alpha_{i,j,k}\, \mathrm{g}_{i,k}^{Q},\ \alpha_{i,j,k} \propto \exp(\mathrm{ReLU}(W\, \mathrm{g}_j^{C})^T\, \mathrm{ReLU}(W\, \mathrm{g}_{i,k}^{Q})), \tag{5}$$

where $\mathrm{g}_{i,k}^{Q}$ is the GloVe embedding for the k-th question word in the i-th question, and g_j^{C} is the GloVe embedding for the j-th context word. The final question-specific context input representation C_i^0 contains: (1) word embeddings, (2) a binary indicator $\mathrm{em}_{i,j}$, whether the j-th context word occurs in the i-th question, and (3) output from the attention.

$$C_i^0 = [c_1; \mathrm{em}_{i,1}; g_{i,1}], \ldots, [c_m; \mathrm{em}_{i,m}; g_{i,m}] \tag{6}$$

Question Integration with QHierRNN Similar to many MC models, contextualized embeddings for the questions are obtained using multiple layers of BiLSTM (we used two layers).

$$\boldsymbol{Q}_i^1 = q_{i,1}^1, \ldots, q_{i,n}^1 = \mathrm{BiLSTM}(\boldsymbol{Q}_i),\ \ \boldsymbol{Q}_i^2 = q_{i,1}^2, \ldots, q_{i,n}^2 = \mathrm{BiLSTM}(\boldsymbol{Q}_i^1) \tag{7}$$

We build a pointer vector for each question to be used in answer prediction layer by first taking a weighted sum of each word vectors in the question.

$$\tilde{q}_i = \sum_{k=1}^{n} \alpha_{i,k} \cdot q_{i,k}^2,\ \alpha_{i,k} \propto \exp(w^T q_{i,k}^2), \tag{8}$$

where w is a trainable vector. We then encode question history hierarchically with LSTMs to generate history-aware question vectors (QHierRNN).

$$p_1, \ldots, p_t = \mathrm{LSTM}(\tilde{q}_i, \ldots, \tilde{q}_t) \tag{9}$$

The answer pointer vectors, $p_1, \ldots, p_t$, will be used in the answer prediction layer.

3.3.2 REASONING

The reasoning component has several IF layers on top of the context encoding, inter-weaved with attention (first on question, then on context itself). We use fully-aware attention (Huang et al., 2018), which concatenates all layers of hidden vectors and uses $S(x, y) = \mathrm{ReLU}(\mathbf{U}x)^T \mathbf{D}\, \mathrm{ReLU}(\mathbf{U}y)$ to compute the attention score between x, y, where $\mathbf{U}, \mathbf{D}$ are trainable parameters and $\mathbf{D}$ is a diagonal matrix. Below we give the details of each layer (from bottom to top).

Integration-Flow $\times 2$ First, we take the question-augmented context representation C_i^0 and pass it to two IF layers.[5]

$$\boldsymbol{C}_i^1 = \text{IF}(\boldsymbol{C}_i^0) \tag{10}$$

$$\boldsymbol{C}_i^2 = \text{IF}(\boldsymbol{C}_i^1) \tag{11}$$

Attention (on Question) After contextualizing the context representation, we perform fully-aware attention on the question for each context words.

$$\hat{q}_{i,j} = \sum_{k=1}^{n} \alpha^{i,j,k} \cdot q_{i,k}^2, \;\; \alpha^{i,j,k} \propto \exp(S([c_i; c_{j,i}^1; c_{j,i}^2], [q_{j,k}; q_{j,k}^1; q_{j,k}^2])) \tag{12}$$

Integration-Flow We concatenate the output from the previous IF layer with the attended question vector, and pass it as an input.

$$\boldsymbol{C}_i^3 = \text{IF}([c_{i,1}^2; \hat{q}_{i,1}], \ldots, [c_{i,m}^2; \hat{q}_{i,m}]) \tag{13}$$

Attention (on Context) We apply fully-aware attention on the context itself (self-attention).

$$\hat{c}_{i,j} = \sum_{k=1}^{m} \alpha^{i,j,k} \cdot c_{j,k}^3, \;\; \alpha^{i,j,k} \propto \exp(S([c_{i,j}^1; c_{i,j}^2, c_{i,j}^3], [c_{k,j}^1; c_{k,j}^2, c_{k,j}^3])) \tag{14}$$

Integration We concatenate the output from the the previous IF layer with the attention vector, and feed it to the last BiLSTM layer.

$$\boldsymbol{C}_i^4 = \text{BiLSTM}([c_{i,1}^3; \hat{c}_{i,1}], \ldots, [c_{i,m}^3; \hat{c}_{i,m}]) \tag{15}$$

3.3.3 Answer Prediction

We use the same answer span selection method (Wang et al., 2017; Wang & Jiang, 2017; Huang et al., 2018) to estimate the start and end probabilities $P_{i,j}^S, P_{i,j}^E$ of the j-th context token for the i-th question.

$$P_{i,j}^S \propto \exp(\left[c_{i,j}^4\right]^T W_S p_i), \quad \tilde{p}_i = \text{GRU}(p_i, \sum_{i,j} P_{i,j}^S c_{i,j}^4), \;\; P_{i,j}^E \propto \exp(\left[c_{i,j}^4\right]^T W_E \tilde{p}_i) \tag{16}$$

To address unanswerable questions, we compute the probability of having no answer:

$$P_i^{\emptyset} \propto \exp\Big(\Big[\sum_{j=1}^{m} c_{i,j}^4; \max_j c_{i,j}^4\Big]^T W p_i\Big). \tag{17}$$

For each question $\boldsymbol{Q}_i$, we first use $P_i^{\emptyset}$ to predict whether it has no answer.[6] If it is answerable, we predict the span to be j^s, j^e with the maximum $P_{i,j^s}^S P_{i,j^e}^E$ subject to the constraint $0 \leq j^e - j^s \leq 15$.

4 Experiments: Conversational Machine Comprehension

In this section, we evaluate FlowQA on recently released conversational MC datasets.

Data and Evaluation Metric We experiment with the QuAC (Choi et al., 2018) and CoQA (Reddy et al., 2018) datasets. While both datasets follow the conversational setting (Section 2.1), QuAC asked crowdworkers to highlight answer spans from the context and CoQA asked for free text as an answer to encourage natural dialog. While this may call for a generation approach, Yatskar (2018) shows that the an extractive approach which can handle Yes/No answers has a high upper-bound – 97.8 F_1. Following this, we apply the extractive approach to CoQA. We handle the Yes/No questions by computing P_i^Y, P_i^N using the same equation for $P_i^{\emptyset}$ (Eq. 17), and find a span in the context for other questions.

[5] We tested different numbers of IF layers and found that the performance was not improved after 2 layers.

[6] The decision threshold is tuned on the development set to maximize the F_1 score.

	Child.	Liter.	Mid-High.	News	Wiki	Reddit	Science	Overall
PGNet (1-ctx)	49.0	43.3	47.5	47.5	45.1	38.6	38.1	44.1
DrQA (1-ctx)	46.7	53.9	54.1	57.8	59.4	45.0	51.0	52.6
DrQA + PGNet (1-ctx)	64.2	63.7	67.1	68.3	71.4	57.8	63.1	65.1
BiDAF++ (3-ctx)	66.5	65.7	70.2	71.6	72.6	60.8	67.1	67.8
FLOWQA (1-Ans)	**73.7**	**71.6**	**76.8**	**79.0**	**80.2**	**67.8**	**76.1**	**75.0**
Human	90.2	88.4	89.8	88.6	89.9	86.7	88.1	88.8

Table 1: Model and human performance (% in F_1 score) on the CoQA test set. (N-ctx) refers to using previous N QA pairs. (N-Ans) refers to providing previous N gold answers.

	F_1	HEQ-Q	HEQ-D
Pretrained InferSent	20.8	10.0	0.0
Logistic Regression	33.9	22.2	0.2
BiDAF++ (0-ctx)	50.2	43.3	2.2
BiDAF++ (1-ctx)	59.0	53.6	3.4
BiDAF++ (2-ctx)	60.1	54.8	4.0
BiDAF++ (3-ctx)	59.5	54.5	4.1
FLOWQA (2-Ans)	**64.1**	**59.6**	**5.8**
Human	80.8	100	100

Table 2: Model and human performance (in %) on the QuAC test set. (baselines from (Choi et al., 2018))

	CoQA	QuAC
Prev. SotA (Yatskar, 2018)	70.4	60.6
FLOWQA (0-Ans)	75.0	59.0
FLOWQA (1-Ans)	**76.2**	64.2
- FLOW	72.5	62.1
- QHierRNN	76.1	64.1
- FLOW - QHierRNN	71.5	61.4
FLOWQA (2-Ans)	76.0	**64.6**
FLOWQA (All-Ans)	75.3	**64.6**

Table 3: Ablation study: model performance on the dev. set of both datasets (in % F_1).

The main evaluation metric is F_1, the harmonic mean of precision and recall at the word level.[7] In CoQA, we report the performance for each context domain (children's story, literature from Project Gutenberg, middle and high school English exams, news articles from CNN, Wikipedia, AI2 Science Questions, Reddit articles) and the overall performance. For QuAC, we use its original evaluation metrics: F_1 and Human Equivalence Score (HEQ). HEQ-Q is the accuracy of each question, where the answer is considered correct when the model's F_1 score is higher than the average human F_1 score. Similarly, HEQ-D is the accuracy of each dialog – it is considered correct if all the questions in the dialog satisfy HEQ.

Comparison Systems We compare FLOWQA with baseline models from CoQA and QuAC. Reddy et al. (2018) presented PGNet (Seq2Seq with copy mechanism), DrQA (Chen et al., 2017) and DrQA+PGNet (PGNet on predictions from DrQA) to address abstractive answer. To incorporate dialog history, CoQA baselines concatenate the most recent previous question and answer to the current question.[8] Choi et al. (2018) used BiDAF++, a strong extractive QA model to QuAC dataset. They concatenate a feature vector encoding turn number to the question embedding and a feature vector encoding previous N answer locations to the context embeddings (denoted as N-ctx). They found this to perform better than just concatenating previous question answer pairs. Yatskar (2018) applied the same model to CoQA by modifying the system to first make a Yes/No decision, and output an answer span only if Yes/No was not selected.

FLOWQA (N-Ans) is our model: similar to BiDAF++ (N-ctx), we concatenate the binary feature vector encoding previous N answer spans to the context embeddings. Here we briefly describe ablated systems: - FLOW removes the flow component from IF layer (Eq. 2-4 in Section 3.2), - QHIERRNN removes hierarchical LSTM on question pointer vectors (Eq. 9 in Section 3.3).

Results Tables 1 and 2 report model performance on CoQA and QuAC, respectively. FLOWQA yields substantial improvement over existing models on both datasets (+7.2% F_1 on CoQA, +4.0% F_1 on QuAC). The larger gain on CoQA, which contains longer dialog chains,[9] suggests that our FLOW architecture can capture long-range conversation history more effectively.

[7] As there are multiple (N) references, the actual score is the average of max F_1 against $N-1$ references. For details, refer to the official eval scripts.

[8] They found concatenating question answer pairs from the further history did not help the performance.

[9] Each QuAC dialog contains 7.2 question/answer pairs on average, while CoQA contains 15 QA pairs.

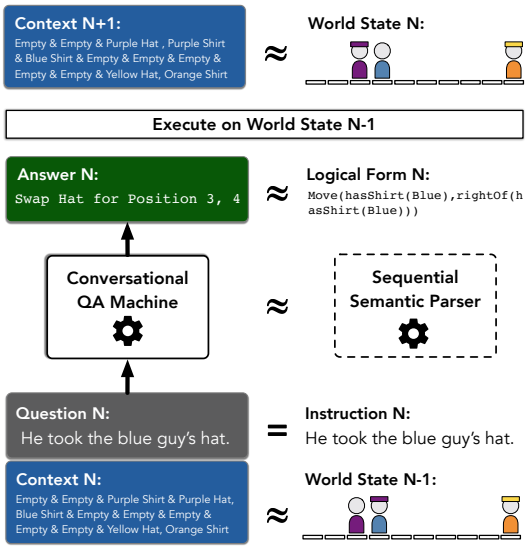

Figure 4: Illustration on reducing sequential instruction understanding to conversational MC. The corresponding units in the semantic parsing approach are shown to the right.

	Sce.	Tan.	Alc.
Suhr & Artzi (2018)	56.1	60.3	71.8
FLOWQA - FLOW	53.0	67.8	82.0
FLOWQA	**64.1**	**74.4**	**84.1**
FLOW Δ	(+11.1)	(+6.6)	(+2.1)

Table 4: Dev accuracy (in %) after all instructions for the three domains in SCONE.

	Sce.	Tan.	Alc.
Long et al. (2016)	14.7	27.6	52.3
Guu et al. (2017)	46.2	37.1	52.9
Suhr & Artzi (2018)	66.4	60.1	62.3
Fried et al. (2018)	72.7	69.6	72.0
FLOWQA - FLOW	58.2	67.9	74.1
FLOWQA	**74.5**	**72.3**	**76.4**
FLOW Δ	(+16.3)	(+4.4)	(+2.3)

Table 5: Test accuracy (in %) after all instructions for the three domains in SCONE.

Table 3 shows the contributions of three components: (1) QHierRNN, hierarchical LSTM for encoding past questions, (2) FLOW, augmenting the intermediate representation from machine reasoning process in the conversation history, and (3) N-Ans, marking the gold answers for N previous turn in the context. We find that FLOW is a critical component. Removing QHierRNN has a minor impact (0.1% on both datasets), while removing FLOW results in a substantial performance drop, with or without using QHierRNN (2-3% on QuAC, 4.1% on CoQA). Without both components, our model performs comparably to the BiDAF++ model (1.0% gain).[10] Our model exploits the entire conversation history while prior models could leverage up to three previous turns.

By comparing 0-Ans and 1-Ans on two datasets, we can see that providing gold answers is more crucial for QuAC. We hypothesize that QuAC contains more open-ended questions with multiple valid answers because the questioner cannot see the text. The semantics of follow-up questions may change based on the answer span selected by the teacher among many valid answer spans, so knowing the exact answer is crucial.

5 EXPERIMENTS: SEQUENTIAL INSTRUCTION UNDERSTANDING

In this section, we consider the task of understanding a sequence of natural language instructions.[11] We reduce this problem to a conversational MC task and apply FLOWQA.

Task Given a sequence of instructions, where the meaning of each instruction may depend on the entire history and world state, the task is to understand the instructions and modify the *world* accordingly. More formally, given the initial world state $\boldsymbol{W}_0$ and a sequence of natural language instructions $\{\boldsymbol{I}_1, \ldots, \boldsymbol{I}_K\}$, the model has to perform the correct sequence of actions on $\boldsymbol{W}_0$, to obtain $\{\boldsymbol{W}_1, \ldots, \boldsymbol{W}_K\}$, the correct world states after each instruction. Fig. 4 gives a simplified example from (Long et al., 2016).

Reducing Sequential Instruction Understanding to Conversational MC We reduce instruction understanding to machine comprehension using the follow mapping. An illustration of this reduction is also in Fig. 4.

- Context $\boldsymbol{C}_i$: We encode the current world state $\boldsymbol{W}_{i-1}$ as a sequence of tokens.

[10] On the SQuAD leaderboard, BiDAF++ outperforms the original FusionNet (Huang et al., 2018) that FLOWQA is based on.

[11] This differs from semantic parsing as we are not mapping to an high-level formal representations but to the world state action, similar to (Suhr & Artzi, 2018).

- Question $\boldsymbol{Q}_i$: We simply treat each natural language instruction $\boldsymbol{I}_i$ as a question.
- Answer $\boldsymbol{A}_i$: We encode the world state change from $\boldsymbol{W}_{i-1}$ to $\boldsymbol{W}_i$ as a sequence of tokens.

At each time step i, the current context $\boldsymbol{C}_i$ and question $\boldsymbol{Q}_i$ are given to the system, which outputs the answer $\boldsymbol{A}_i$. We simplified FLOWQA to prevent overfitting.[12] Appendix A.2 contains the details on model simplification and reduction rules, i.e., mapping from the world state and state change to a sequence of token.

We also encode history explicitly by concatenating preceding questions with the current one and by marking previous answers in the current context similar to N-Ans in conversational MC. During training, the previous gold answer (i.e., the world state change after each previous instruction) is given to the model. In testing, the model predictions to the previous instructions are used.

5.1 RESULTS

We evaluate our model on the sequential instruction understanding dataset SCONE (Long et al., 2016), which contains three domains (SCENE, TANGRAMS, ALCHEMY). Each domain has a different environment setting (see Appendix A.2). We compare our approaches with prior works (Long et al., 2016; Guu et al., 2017; Suhr & Artzi, 2018; Fried et al., 2018), which are semantic parsers that map each instruction into a logical form, and then execute the logical form to update the world state. The model performance is evaluated by the correctness of the final world state after five instructions.

The development and test set results are reported in Tables 4 and 5. Even without FLOW, our model (FLOWQA-FLOW) achieves comparable results in two domains (Tangrams and Alchemy) since we still encode the history explicitly. When augmented with FLOW, our FLOWQA model gains decent improvements and outperforms the state-of-the-art models for all three domains.

6 RELATED WORK

Sequential question answering has been studied in the knowledge base setting (Iyyer et al., 2017; Saha et al., 2018; Talmor & Berant, 2018), often framed as a semantic parsing problem. Recent datasets (Choi et al., 2018; Reddy et al., 2018; Elgohary et al., 2018; Saeidi et al., 2018) enabled studying it in the textual setting, where the information source used to answer questions is a given article. Existing approaches attempted on these datasets are often extensions of strong single-turn models, such as BiDAF (Seo et al., 2016) and DrQA (Chen et al., 2017), with some manipulation of the input. In contrast, we propose a new architecture suitable for multi-turn MC tasks by passing the hidden model representations of preceding questions using the FLOW design.

Dialogue response generation requires reasoning about the conversation history as in conversational MC. This has been studied in social chit-chats (e.g., Ritter et al., 2011; Li et al., 2017; Ghazvininejad et al., 2018) and goal-oriented dialogs (e.g., Bordes & Weston, 2016; Lewis et al., 2017). Prior work also modeled hierarchical representation of the conversation history (Park et al., 2018; Suhr & Artzi, 2018). While these tasks target reasoning with the knowledge base or exclusively on the conversation history, the main challenge in conversational MC lies in reasoning about *context* based on the conversation history, which is the main focus in our work.

7 CONCLUSION

We presented a novel FLOW component for conversational machine comprehension. By applying FLOW to a state-of-the-art machine comprehension model, our FLOWQA is able to encode the conversation history more comprehensively, and thus yields better performance. When evaluated on two recently proposed conversational challenge datasets and three domains of a sequential instruction understanding task (through reduction), FLOWQA outperforms *all* current best models.

While our approach provides a substantial performance gain, there is still room for improvement. In the future, we would like to investigate even more efficient and fine-grained ways to model the conversation flow, as well as methods that enable machines to engage an more active and natural conversational behaviors, such as asking clarification questions.

[12] This domain contains substantially smaller training data ($\sim$20K QA pairs).

REFERENCES

Antoine Bordes and Jason Weston. Learning end-to-end goal-oriented dialog. *CoRR*, abs/1605.07683, 2016.

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

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. QuAC : Question answering in context. *EMNLP*, 2018.

Ahmed Elgohary, Chen Zhao, and Jordan Boyd-Garber. A dataset and baselines for sequential open-domain question answering. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Daniel Fried, Jacob Andreas, and Dan Klein. Unified pragmatic models for generating and following instructions. *NAACL*, 2018.

Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *NIPS*, 2016.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, William B. Dolan, Jianfeng Gao, Wen tau Yih, and Michel Galley. A knowledge-grounded neural conversation model. *AAAI Conference on Artificial Intelligence (AAAI)*, abs/1702.01932, 2018.

Kelvin Guu, Panupong Pasupat, Evan Zheran Liu, and Percy Liang. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. *ACL*, 2017.

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

Hsin-Yuan Huang, Chenguang Zhu, Yelong Shen, and Weizhu Chen. FusionNet: Fusing via fully-aware attention with application to machine comprehension. *ICLR*, 2018.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1821–1831, Vancouver, Canada, July 2017. Association for Computational Linguistics.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Mike Lewis, Denis Yarats, Yann Dauphin, Devi Parikh, and Dhruv Batra. Deal or no Deal? End-to-end learning for negotiation dialogues. *CoRR*, abs/1706.05125, 2017.

Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Daniel Jurafsky. Adversarial learning for neural dialogue generation. In *EMNLP*, 2017.

Reginald Long, Panupong Pasupat, and Percy Liang. Simpler context-dependent logical forms via model projections. In *ACL*, 2016.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pp. 6297–6308, 2017.

Yookoon Park, Jaemin Cho, and Gunhee Kim. A hierarchical latent structure for variational conversation modeling. In *NAACL-HLT*, 2018.

Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *EMNLP*, 2014.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *NAACL*, 2018.

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

Alan Ritter, Colin Cherry, and William B. Dolan. Data-driven response generation in social media. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2011.

Marzieh Saeidi, Max Bartolo, Patrick Lewis, Sameer Signh, Tim Rocktschel, Mike Sheldon, Guillaume Bouchard, and Sebastian Riedel. Interpretation of natural language rules in conversational machine reading. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Amrita Saha, Vardaan Pahuja, Mitesh M Khapra, Karthik Sankaranarayanan, and Sarath Chandar. Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *ICLR*, 2016.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 2014.

Alane Suhr and Yoav Artzi. Situated mapping of sequential instructions to actions with single-step reward observation. In *ACL*, 2018.

A. Talmor and J. Berant. The web as knowledge-base for answering complex questions. In *NAACL*, 2018.

Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. *ICLR*, 2017.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. Gated self-matching networks for reading comprehension and question answering. In *ACL*, 2017.

Mark Yatskar. A qualitative comparison of CoQA, SQuAD 2.0 and QuAC. In *ArXiv*, 2018.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. QANet: Combining local convolution with global self-attention for reading comprehension. *ICLR*, 2018.

A Implementation Details

A.1 Conversational Question Answering

We make use of spaCy for tokenization. We additionally fine-tuned the GloVe embeddings of the top 1000 frequent question words. During training, we use a dropout rate of 0.4 (Srivastava et al., 2014) after the embedding layer (GloVe, CoVe and ELMo) and before applying any linear transformation. In particular, we share the dropout mask when the model parameter is shared, which is also known as variational dropout (Gal & Ghahramani, 2016). We batch the dialogs rather than individual questions. The batch size is set to one dialog for CoQA (since there can be as much as 20+ questions in each dialog), and three dialog for QuAC (since the question number is smaller). The optimizer is Adamax (Kingma & Ba, 2015) with a learning rate $\alpha = 0.002$, $\beta = (0.9, 0.999)$ and $\epsilon = 10^{-8}$. A fixed random seed is used across all experiments. All models are implemented in PyTorch (`http://pytorch.org/`). We use a maximum of 20 epochs, with each epochs passing through the data once. It roughly takes 10 to 20 epochs to converge.

A.2 Sequential Semantic Parsing

We begin by elaborating the simplification for FlowQA for the sequential semantic parsing task. First, we use the 100-dim GloVe embedding instead of the 300-dim GloVe and we did not use any contextualized word embedding. The GloVe embedding is fixed throughout the training. Secondly, the embedding for tokens in the context $\boldsymbol{C}$ are trained from scratch, since $\boldsymbol{C}$ consists of synthetic tokens. Also, we removed word-level attention because the tokens in contexts and questions are very different (one is synthetic, while the other is natural language). Additionally, we removed self-attention since we find it unhelpful in this reduced QA setting (we speculate it's because the context here is usually very short). We use the same hidden size for both integration LSTMs and Flow

GRUs. But we tune the hidden size for the three domains independently. In SCENE, we use hidden size of $h = 100$. In TANGRAMS, we use $h = 75$. And in ALCHEMY, we use $h = 50$. We also batch by dialog and use a batch size of 8. A dropout rate of 0.3 is used and is applied before every linear transformations.

Environment for the Three Domains In SCENE, each environment has ten positions with at most one person at each position. The domain covers four actions (enter, leave, move, and trade-hats) and two properties (hat color, shirt color). In TANGRAMS, the environment is a list containing at most five shapes. This domain contains three actions (add, move, swap) and one property (shape). Lastly, in ALCHEMY, each environment is seven numbered beakers and covers three actions (pour, drain, mix) dealing with two properties (color, amount).

Reducing World State to Context Now, we give details on the encoding of context from the world state. In SCENE, there are ten positions. For each position, there could be a person with shirt and hat, a person with a shirt, or no person. We encode each position as two integers, one for shirt and one for hat. (so the context length is ten) Both integer take the value that corresponds to being a color or being empty. In TANGRAMS, originally there are five images. But some commands could reduce the number of images or bring back removed images. Since the number of images present is no greater than five, we always have five positions available (so the context length is five). Each position consists of an integer, representing the ID of the image, and a binary feature. Every time an image is removed, we append it at the back. The binary feature is used to indicate if the image is still present or not. In ALCHEMY, there are always seven beakers. So the context length is seven. Each position consists of two numbers, the color of the liquid at the top unit and the number of units in the beaker. An embedding layer is used to turn each integers into a 10-dim vector.

Reducing the Logical Form to Answer Next, we encode the change of world state (i.e., the answer) into four integers. The first integer is the type of action that's performed. The second and third integer represent the position of the context, which the action is acted upon. And the fourth integer represents additional property for action performed. For example, in the ALCHEMY domain, we use $(0, i, j, 2)$ to mean "pour 2 units of liquid from beaker i to beaker j", and $(1, i, i, 3)$ to mean "throw out 3 units of liquid in beaker i".