Flexibly Focusing on Supporting Facts, Using Bridge Links, and Jointly Training Specialized Modules for Multi-hop Question Answering

Tareq Alkhaldi, Chenhui Chu and Sadao Kurohashi

***Abstract*—With the help of the detailed annotated question answering dataset HotpotQA, recent question answering models are trained to justify their predicted answers with supporting facts from context documents. Some related works train the same model to find supporting facts and answers jointly without having specialized models for each task. The others train separate models for each task, but do not use supporting facts effectively to find the answer; they either use only the predicted sentences and ignore the remaining context, or do not use them at all. Furthermore, while complex graph-based models consider the bridge/connection between documents in the multi-hop setting, simple BERT-based models usually drop it. We propose Flexible-FocusedReader (FFReader), a model that 1) Flexibly focuses on predicted supporting facts (SFs) without ignoring the important remaining context, 2) Focuses on the bridge between documents, despite not using graph architectures, and 3) Jointly learns predicting SFs and answering with two specialized models. Our model achieves consistent improvement over the baseline. In particular, we find that flexibly focusing on SFs is important, rather than ignoring remaining context or not using SFs at all for finding the answer. We also find that tagging the entity that links the documents at hand is very beneficial. Finally, we show that joint training is crucial for FFReader.**

***Index Terms*—Bridge links, joint training, multi-hop question answering, supporting facts, transformer.**

I. Introduction

THE task of question answering (QA) is to find an answer to a natural language question from a given text [1]. With the goal of training systems to apply reasoning and inference on text, and measuring their performance quantitatively, many datasets have been introduced. One of the early large-scale ones include SQuAD [1], where questions were designed to be answered from a single paragraph, and thus called single-hop QA. Systems achieved human performance, without achieving the sought-after reasoning skill [2] [3], as questions could mostly be answered from a single sentence which encouraged the systems to focus more on matching information between the question and text [4].

To stimulate models to use more complex reasoning, the task of multi-hop QA was introduced. In this task, reasoning over multiple documents is required to find an answer [4]. A popular dataset for this task is HotpotQA [3] which, in addition to the answer, has per sentence annotations for which sentences are supporting facts (SFs). The goal of asking models to predict answers and supporting facts is to encourage models to explain how they reach to an answer.

Everyone is with Kyoto University, Japan (e-mail: {alkhaldi,chu,kuro }@nlp.ist.i.kyoto-u.ac.jp).

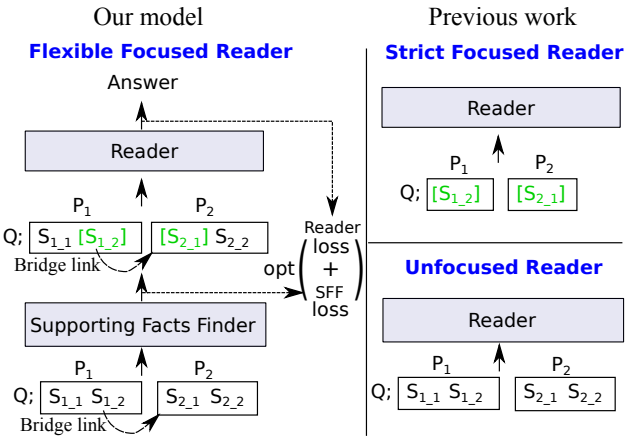


Fig. 1. High-level comparison between our proposed model with similar non graph-based related work models. With a question and a paragraph pair as input, our model uses SFs annotations effectively and does not ignore the bridge link between the paragraphs. The reader module is jointly trained with the SFF module. Here, S_{j_i} represents sentence i of paragraph j. Strict Focused Reader means that the model only uses the predicted SFs as input to the reader, while Unfocused Reader does not use SFs to predict the answer.

Models that are recently introduced for this task generally divide into two architectures: Graph-based models that use some form of Graph Neural Networks (GNNs) [5] like [6]–[8], and non graph-based models that are a pipeline of BERT-based [9] models like [10], [11]. Some models use predicted answers to find the SFs [8], [12], or use SFs prediction as a second task on the same model that predicts the answers [6], [13] (not using any predicted SFs as input to find the answer; i.e. Unfocused Reader), or only feed the predicted SFs to the answering model, ignoring the remaining context [10], [14] (i.e. Strict Focused Reader).

We think that the healthy way to find an answer is by reasoning on the SFs, however ignoring the remaining context might leave out important information, either due to annotation problems or sub-optimal SFs prediction. It will also limit the model to answer only from within the SFs. We propose a novel way of focusing on the SFs; we tag them while keeping the remaining context, thus allowing our reader model to predict correct answers even from outside of the SFs as we show in Sec. VII.

We also identify and tag entities that link paragraph pairs, as we find it is an important cue that non graph-based models

lack. Models incorporate complex[1] GNNs to include this link, while we keep our model simple and still include it.

In order to benefit from both sub-task signals, previous models [6], [7], [11] have jointly trained the same model to both answer and predict SFs. As observed by Beltagy et al., [11], this training method hurts the reader's performance, likely due to less capacity and to not being specialized in each sub-task. To avoid this, we train a specialized reader and a SF finder modules jointly.

We show a comparison of our work and previous work in Figure 1. Our final model achieves clear improvement over the baseline and non graph-based models, and achieves comparable results with the more complex state-of-the-art models. We show that when jointly trained, flexibly focusing on SFs benefits from the SFs annotations for finding the answer without limiting the answer space. Our proposal of tagging SFs to focus on them (instead of only using them or ignoring them at all) is a general recommendation for QA tasks, and it can be applied even to datasets that do not have SFs supervision like SQUAD [1]; we can still predict SFs and use them.[2] Furthermore, we show that identifying and tagging bridge entities is important for multi-hop QA, and we expect this finding to be applicable to any other Wikipedia-based multi-hop QA dataset where we can benefit from the hyperlinks between articles like WikiHop [4] and HybridQA [15].

II. Related Work

A. Single-hop QA

Questions in single-hop QA datasets like SQuAD [1], WebQuestions [16], SimpleQuestions [17] and NaturalQuestions [18] can be answered using a single paragraph or document as context. Since the introduction of Transformers [19] and BERT [9], best performing systems have been extensions of such pre-trained models like RoBERTa [20], ALBERT [21] and ELECTRA [22], with a typical span prediction head on top. The task of mere span prediction from a single document was not enough to let the models learn to answer more complex questions that require reasoning across multiple documents, and so multi-hop QA was considered as a next step.

B. Multi-hop QA on Knowledge Bases

Some datasets focus on enabling QA over knowledge bases (KBs). The WebQuestions semantic parses (WebQSP) dataset [23] provides semantic parses of questions answerable from Freebase [24] and they require up to 2-hop reasoning. ComplexWebQuestions (CWQ) [25] provides crowd-sourced compositional natural language questions answerable from Freebase, and also requiring multi-hop reasoning. Some models for multi-hop knowledge base QA (KBQA) decompose complex questions into simpler questions and do reasoning depending on intermediate answers [25] or use two networks for finding answers from the KB and deciding better intermediate reasoning [26]. Knowledge bases however can be hard to maintain and noisy to generate automatically.

C. Multi-hop QA on Text

Multi-hop QA over text datasets like HotpotQA [3] and WikiHop [4] were introduced to encourage systems to learn more complex reasoning as the pieces of evidence to answer a question are scattered among different documents, as opposed to single-hop QA datasets and KB-based datasets. HotpotQA also includes SFs annotations to encourage models to explain their reasoning.

1) Graph vs Non-graph Based Models: Multi-hop QA requires the model to hop between documents that are usually connected by a link to find the answer. This bridge connection is ignored by non graph-based models [10], [11] where they encode the concatenated question and context, and perform classifications on top of the transformers [27] output. To make use of the connection between sentences and documents, some models incorporate GNNs [28], [29]. SAE [7] predicts answers directly from transformers output, but applies GNNs on top of the generated sentence embeddings to predict SFs. HGN [6] constructs a hierarchical graph of multiple levels of granularity (paragraphs, sentences and entities), and predicts answers and SFs on top of this GNN. Shao et al., [8] argue that graph structure might not be necessary for multi-hop question answering when pre-trained transformers are fine-tuned, and that graph attention can be considered as a special case of self-attention. In our model, we follow the simplicity of Longformer [11] but without sacrificing the bridge connection signal between documents. BigBird [30] is a very similar model to Longformer, with the main difference being the addition of either random attention or external tokens for global attention.

2) Utilization of Supporting Facts Annotations: HGN and DFGN [13] models use SFs annotations implicitly in a multi-task setting. QUARK [14] and TAP2 [10] explicitly use predicted SFs as the only context available to the answer finding model. This strictness is harmful as not only the accuracy of the SF prediction is not optimal, but also the golden annotation itself has the problem of leaving out important related sentences, as discussed in Sec. VII-J. We circumvent this by tagging the predicted SFs while keeping the remaining context.

3) Joint Training: Models that apply joint training like HGN, DFGN, QFE [31] and Longformer 1-stage version, do it in a multi-task way on the same model. The official HotpotQA baseline model [3] adds several layers on top of its SFs prediction module but its model still shares the same low-level representations, therefore joint training still happens on the same model. As Longformer results show, 2-stage mode (2 separate models for answer and SFs prediction) is better than 1-stage mode, because having separate models for each task means more capacity and specialized models. In our model, we jointly train separate specialized models. We also experiment with additional settings like different ratios of loss combination and pre-initialization for joint fine-tuning as discussed in Sec. VII-G and VII-F.

[1]Complexity can be in graph construction and keeping information of nodes and edges for every example (which includes recognizing named entities in some models like [6]), or in performance, where the run overhead depends on the number of nodes/edges, and the number of message passing iterations

[2]The investigation of such application is left for future work.

TABLE I
STATISTICS SHOWING THE AMOUNT OF QUESTION TYPES FOR EACH DATA SPLIT IN HOTPOTQA. NOTE THAT THE TEST SPLIT IS HIDDEN IN THE DISTRACTOR SETTING.

Question Type	train	dev
Comparison	17,456	1,487
Bridge	72,991	5,918
Total	90,447	7,405

III. THE HOTPOTQA DISTRACTOR TASK

A. Task Description

We use the HotpotQA dataset [3] which has two settings: Distractor and fullwiki. In this paper, we focus on the distractor setting as it is only concerned with the reader model part of the problem, not the information retrieval part. In the distractor setting, 10 paragraphs from 10 different Wikipedia documents are given, only 2 paragraphs are related to the question to be answered and explained. The two sub-tasks are: 1) Answer prediction, and 2) Supporting facts prediction. They are evaluated with exact match (EM) and partial match (F1) metrics. The final performance is evaluated with a joint EM and F1 score.

The questions in this dataset have two types: "Bridge" and "Comparison." "Bridge" questions are anchored around a bridge entity (i.e., a hyperlink) that connects the paragraphs, while "Comparison" questions are about two paragraphs, not necessarily connected by a link. Table I shows statistics about the percentage of each question type in the dataset, and we see that "Bridge" type questions are the majority.

B. Data Preparation

Each question with its 2 gold and 8 distractor paragraphs is considered an example. In training, we generate 3 paragraph pairs for each example: 2 gold, 1 gold 1 distractor, and 2 distractor paragraphs. In evaluation, we consider all possible paragraph pairs, and we choose the pair with the highest score as shown in Sec. IV-D.

As alternative preparation settings, we also experiment with only using 2 gold paragraphs, or with 2 gold and 1 gold and 1 distractor without 2 distractor paragraphs, but we find that the performance degrades in both cases. Related work methods are either not concretely explained or not applicable to our model; Longformer [11] inputs 10 paragraphs at once, while HGN [6] selects several paragraphs using string matching heuristics together with a trained ranker when needed, and uses them for training. SAE [7] uses only gold paragraphs, and uses a trained ranker to retrieve top 2 paragraphs.

IV. MODEL FLOW

We show an overview of our model in Fig. 2. We first explain the model flow, then talk about our contributions in Sec. V.

A. LongRoBERTa

The basic unit in our model is a long version of RoBERTa [20], which is constructed in a similar way to Longformer, with the difference being in the maximum token length. LongRoBERTa and Longformer are different from plain RoBERTa mainly because of using global attention, which we set only on selected tokens as explained in Longformer [11]. Following Longformer authors' instructions of continuing pretraining after the construction of a longer RoBERTa, we pretrained on Wikitext103 [32] for 3k steps. Longformer inputs 10 paragraphs at once as context with maximum token length of 4,096. This can be noisy as paragraphs from different documents are not coherent, and it is difficult to focus on links between paragraphs as there can be many links. Therefore, we only consider 2 paragraphs at a time with a maximum token length of 1,024, reducing noise and allowing us to do bridge tagging. LongRoBERTa is used for the reader module and the SFs finder module. We do not use the plain 512 tokens RoBERTa to avoid truncating or complex splitting into windows. We show the effects of using a plain RoBERTa versus LongRoBERTa in Sec. VII-H.

B. Supporting Facts Finder (SFF) Module

We concatenate the question with the paragraph pair after tagging with special tokens as follows: "`[CLS] [q] question [/q] [t] title1 [/t] sent1_1 [/s] sent1_2 [/s]...[t] title2 [/t] sent2_1 [/s]...[SEP]`" where special tokens `[q]` and `[/q]` are question boundaries, `[t]` and `[/t]` are paragraph title boundaries, and `[/s]` is sentence ends. We consider `[t]` to represent the whole paragraph, and `[/s]` to represent the sentence, and we only assign `[t]` and `[/s]` tokens to have global attention, all following Longformer. Also similar to Longformer, we apply two-layer feedforward networks on top of the LongRoBERTa output of `[t]` and `[/s]` tokens to calculate the binary scores of the relatedness of paragraph *j* ($\mathbf{o}_{para\_j}$) and sentence *i* of paragraph *j* ($\mathbf{o}_{sent\_j\_i}$):

$$\mathbf{o}_{para\_j} = MLP_1(\mathbf{P}_j) \tag{1}$$

$$\mathbf{o}_{sent\_j\_i} = MLP_2(\mathbf{S}_{j\_i}) \tag{2}$$

where MLP($\cdot$) denotes Multi Layer Perceptron (MLP), $\mathbf{P}_j$ is the embedded output of token `[t]` of paragraph j, and $\mathbf{S}_{j\_i}$ is the embedded output of token `[/s]` of sentence *i* of paragraph *j*. From the two scores in $\mathbf{o}_{para\_j}$ (related and not related), we denote P_j to be the logit of paragraph j being related to the question. We use cross entropy to get the final loss of this module, SFF_{loss}:

$$SFF_{loss} = CE(\mathbf{o}_{para}, \mathbf{y}_{para}) + CE(\mathbf{o}_{sent}, \mathbf{y}_{sent})$$

where $\mathbf{o}_{para}$ and $\mathbf{o}_{sent}$ are vectors of binary scores for every paragraph and sentence, $\mathbf{y}_{para}$ and $\mathbf{y}_{sent}$ are the labels of the paragraph being related or sentence being a supporting fact, respectively. CE() represents cross entropy loss function. Note that in $\mathbf{y}_{para}$, a paragraph is given label `1` if it contains at least 1 SF.

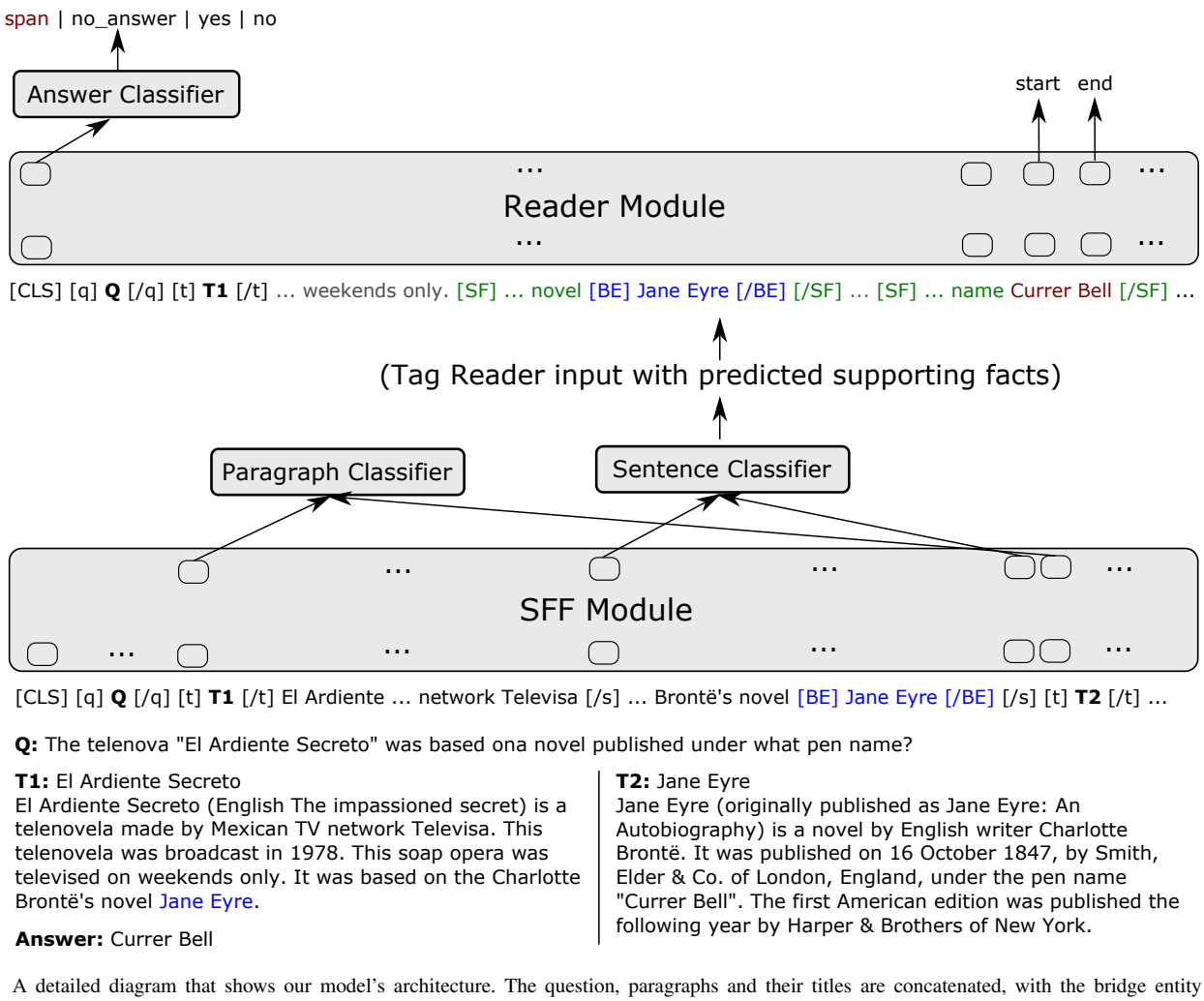


Fig. 2. A detailed diagram that shows our model's architecture. The question, paragraphs and their titles are concatenated, with the bridge entity tagged with [BE] tag (tokens are shown in blue color). The [t] tags represent the paragraphs and are passed to "Paragraph Classifier" to calculate the score P_j of paragraph j as in Eq. (4). The [/s] tags represent the sentences and are passed to the "Sentence Classifier" that classifies each sentence as a SF or not. In the input to the reader module, the sentences that are predicted to be SFs are tagged with [SF] tags (tokens are shown in green color). The start and end of a candidate answer span is then predicted, along with a classifier on the [CLS] token deciding whether to use this predicted span (tokens are shown in red color), give a yes/no answer or decide that no answer is available. Note that Joint training is not depicted in this figure for clarity.

C. Reader Module

We pass the prepared input without `[/s]` tokens to another instance of LongRoBERTa, with global attention for all question tokens only. Following a typical QA model [9], we predict the start and end tokens from context outputs, and apply a multi-class classifier on top of the embedded output of the `[CLS]` token ($\mathbf{H}_{CLS}$) with 4 classes as:

$$\mathbf{o}^i_{answer} = \mathbf{W}_a \mathbf{H}_{CLS}, \tag{3}$$

where $\mathbf{W}_a$ is a learnable weight matrix, and for paragraph pair i, $\mathbf{o}^i_{answer} \in \mathbb{R}^{1\times 4}$ represents the logits of the 4 classes: [$span$, no_ans, yes, no], where they mean the answer is the span predicted by start and end logits, or no answer is found in the current paragraph pair, or yes and no answers to "Comparison" type questions, following Asai et al. [12]. The final loss, $Reader_{loss}$ is calculated as:

$$\begin{aligned} Reader_{loss} = & \, CE(\mathbf{o}_{start}, \mathbf{y}_{start}) \\ & + CE(\mathbf{o}_{end}, \mathbf{y}_{end}) \\ & + CE(\mathbf{o}_{answer}, \mathbf{y}_{answer}) \end{aligned}$$

where $\mathbf{o}_{start}$ and $\mathbf{o}_{end}$ are the logits of the start and end positions of the predicted span in the range of all possible indices. $\mathbf{y}_{start}$, $\mathbf{y}_{end}$ and $\mathbf{y}_{answer}$ are the labels of the start, end positions, and the answer class, respectively.

D. Evaluation Time Paragraph Pair Selection

In evaluation time, we select the paragraph pair i that has the highest score $Pair_i$ as follows:

$$Pair_i = \sum_{j \in S_i} P_j - no\_ans^i_{logit} \tag{4}$$

where S_i is the set of the selected paragraphs in pair i according to the binary score defined in Eq. (1), $no\_ans^i_{logit}$ is the no answer logit for paragraph pair i as explained in Sec. IV-C.

V. Our Proposal

A. Overview

With the flow described in Sec. IV, we add our contributions that manifest in 1) Flexibly focusing on SFs, that are usually not used in predicting the answer, or used in a strict, non-flexible way, 2) Tagging bridge entities ("Jane Eyre" in the example) that are usually ignored with non graph based systems, and 3) Joint training the SFF module with the reader module which proves to be crucial for the flexibly focusing on SFs. We explain each contribution in detail in the following sections.

B. Flexibly Focusing on Predicted SFs

When preparing the input for the reader module (Sec. IV-C), we focus on predicted SFs by tagging related sentences as "`[SF] sent1_2 [/SF]`," while keeping the tagged bridge entities. We call our model that uses this technique: "FlexibleFocusedReader" (FFReader). We consider the baseline to be "UnfocusedReader" where it is similar to our model except that it does not tag any SFs. We also compare against "StrictFocusedReader" (SFReader) where only predicted SFs are included as context, ignoring remaining sentences.

C. Bridge Entity Tagging (BET)

We use the raw Wikipedia text with hyperlinks to extract links and their indices and save everything in an indexed database. For each paragraph in the input pair, we retrieve available links and we match them against the other paragraph in the pair. If the paragraphs are linked, we tag the tokens of the bridge entity as `[BE] hyperlink text [/BE]`. This tagging is important mostly in "Bridge" question types. Since they are the majority (as shown in Sec. III-A), the importance of this proposal is well reflected in practice (Sec. VII).

D. Joint Training

We jointly train the SFF and reader modules by combining their losses as follows:

$$Loss = SFF_{loss} * \lambda + Reader_{loss} * (1 - \lambda) \qquad (5)$$

where $\lambda \in [0, 1]$ is a hyperparameter to control the importance of each module. We find in our experiments that the optimal λ is 0.5. Detailed λ comparison can be found in Sec. VII-G. While this joint training is crucial for the reader module, we find that it does not improve the SFF module. In fact, it degrades its performance, therefore, in our final model we use a separately trained SFF module + the jointly trained reader module. We also experiment with a joint training setting where we initialize the modules with separately trained ones and only fine-tune them, but we did not gain major improvement. We give more details in Sec. VII-F.

TABLE II
Test scores on the distractor setting of HotpotQA. We split the top models in the leaderboard into categories based on their architectures. **Ours** denotes our FFReader-large model. Models with † sign lack any details other than the test scores on the official leadereboard (https://hotpotqa.github.io/) as of January 28th, 2021.

Model Categories	Ans		Sup		Joint	
	EM	F1	EM	F1	EM	F1
Non graph-based						
TAP2 [10]	64.99	78.59	55.47	85.57	39.77	69.12
Longformer [11]	68.00	81.25	63.09	88.34	45.91	73.16
ETC-large[3] [30]	68.12	81.18	**63.25**	**89.09**	**46.40**	73.62
FFReader-large (Ours)	**68.89**	**82.16**	62.10	88.42	45.61	**73.78**
Graph-based						
SAE-large [7]	66.92	79.62	61.53	86.86	45.36	71.45
SEGraph †	68.03	81.17	61.70	87.43	44.86	72.40
C2F Reader [8]	67.98	81.24	60.81	87.63	44.67	72.73
BFR-Graph †	**70.06**	**82.20**	61.33	88.41	45.92	74.13
HGN-large [6]	69.22	82.19	**62.76**	**88.47**	**47.11**	**74.21**
Unknown arch.						
GSAN-large †	68.57	81.62	62.36	88.73	46.06	73.89
SpiderNet-large †	70.15	83.02	**63.82**	**88.85**	47.54	74.88
AMGN+ †	**70.53**	**83.37**	63.57	88.83	**47.77**	**75.24**

VI. Experiments

A. Implementation Details

We use Transformers library [19] in our implementation with *base* and *large* versions of RoBERTa [20]. Both *base* and *large* versions are extended to have max tokens of 1,024 (originally 512) following the instructions of Beltagy et al., [11] for building "long" version of pre-trained models. The difference between them is: 12-layer, 768-hidden and 12-heads for *base* vs 24-layer, 1,024-hidden and 12-heads for *large*. For both *base* and *large*, we use the Adam optimizer with warmup equal to 0.1 of the total steps and linear decay. We use a learning rate of 5e-5 with a batch size of 32. For *base* joint experiments, we use a learning rate of 2e-4 with a 512 batch size with Adasum [33]. We train *base* versions for 10 epochs, while *large* for 20 epochs, and use λ = 0.5 for both versions of joint experiments.

B. Results

We show the test results of our *large* model on the distractor setting of the HotpotQA dataset in Table II. "Ans" and "Sup" refer to the tasks of finding answers and supporting facts, respectively. "Joint" refers to the joint evaluation where for each question, the answer *and* the supporting facts should be correct to get the EM score, or partially correct to get the F1 score. We see that our model outperforms all non graph-based models and several graph-based ones for "Ans" task, and achieves comparable scores with sophisticated graph-based models on both tasks.

We think the reason Longformer and BigBird-ETC have better EM scores (and F1 for BigBird-ETC) in predicting SFs in the "Sup" task is probably because when they concatenate all 10 paragraphs, they have a lot of negative sentences to look

[3]ETC-large is the name reported on the official leaderboard, but the actual full name is BigBird-ETC.

TABLE III
COMPARISON BETWEEN DEV SCORES FOR "ANS" TASK OF DIFFERENT TYPES OF *base* READER MODULES AS DESCRIBED IN SEC. V-B. WE SPLIT THE SCORES INTO (B/C) MEANING THE RELATIVE SCORES FOR (BRIDGE/COMPARISON) QUESTION TYPES. FFREADER WITH BET AND JOINT TRAINING IS OUR PROPOSAL.

Model	No BET		BET	
	EM (B/C)	F1 (B/C)	EM (B/C)	F1 (B/C)
Unfocused	63.02 (61.47/69.22)	76.96 (77.40/75.20)	63.84 (62.93/67.43)	77.80 (78.80/73.75)
SFReader	60.85 (60.31/62.99)	74.51 (75.85/69.17)	61.78 (61.06/64.64)	75.20 (76.41/70.36)
FFReader	61.00 (60.42/63.29)	74.93 (76.41/69.02)	61.80 (61.42/63.29)	75.79 (77.47/69.17)
+ joint train	63.62 (62.05/69.87)	77.73 (78.19/75.89)	**64.80** (63.60/69.62)	**78.46** (79.18/75.59)
+ gold SFs	65.55 (64.40/70.17)	79.36 (80.32/75.49)	66.71 (65.90/69.92)	80.14 (81.19/75.94)

TABLE IV
WE SHOW THE EFFECT OF BET AND JOINT TRAINING ON THE *base* SFF MODULE'S DEV SCORES FOR THE "SUP" TASK. THE USE OF GOLD PARAGRAPHS (EXCLUDING PARAGRAPH PAIR SELECTION) SHOWS THE UPPER-BOUND OF SFS PREDICTION. WE SPLIT THE SCORES INTO (B/C) MEANING THE RELATIVE SCORES FOR (BRIDGE/COMPARISON) QUESTION TYPES.

Model	No BET		BET	
	EM (B/C)	F1 (B/C)	EM (B/C)	F1 (B/C)
SFF module	59.00 (55.73/72.01)	85.85 (84.81/89.99)	**61.04** (58.78/69.87)	**87.16** (86.44/89.99)
+ joint train	60.46 (57.73/71.26)	86.87 (85.89/90.78)	60.20 (57.67/70.27)	87.08 (86.36/89.94)
+ gold para	61.58 (58.82/72.56)	87.76 (87.10/90.38)	62.86 (60.85/70.86)	88.52 (88.04/90.43)

at and train on. One possible future work is to experiment with even more negative sampling settings than discussed in Sec. III-B.

In the "Sup" task in Table II, EM and F1 scores represent precision and recall of SFs, respectively. Even though in the training data, we make sure the answer is in the gold SFs, our FFReader is flexible and does not require the answer to be in the predicted SFs, as we show in Table XI- Question 2. Therefore, we do not suffer from the necessity of having a huge recall on the SFs to make sure the answer is included, as in an SFReader. For example, the SFReader-based model TAP2 [10] selects SFs using a fixed threshold to make sure their recall (F1 score) is high since it is more important than precision (EM score), because if the SF containing the answer is missed, there is no way to answer correctly.

In our SFF module, we do not actually explicitly control the threshold for selecting a SF, we follow Longformer by predicting a binary score for each sentence [score_0, score_1]; if score_1 is larger than score_0, the sentence is considered a SF. Therefore, there is no precision/recall balance hyper-parameter that we tune.

Related work of graph-based architectures usually apply Graph Neural Networks (GNNs) [29] on top of a Transformers [27] model to do SFs prediction and answering. To use our flexible focusing on SFs proposal, they need to first identify the SFs, then answer. Therefore they need two separate versions of their model for each sub-task. This would mean their final architecture would be as follows: "(Transformer model + GNNs) → SFs + (Transformer model + GNNs) → Answer." Our model however would just be "(Transformer model) → SFs + (Transformer model) → Answer." In this sense, the double addition of GNNs is an added complexity and performance cost.[4]

We detail the sources of our improvement through an ablation study on the *base* version of our model. The ablation was done using the base version instead of the large version of RoBERTa because of computation and time constraints. The base version fits into our smaller, more available GPUs, while the large version needs longer time on our limited number of larger GPUs.

[4]Adding GNNs on top of Transformers is orthogonal to our work, and can still be added to close the gap between our model and graph-based models.

VII. DISCUSSION

A. Reader Module

In Table III, we compare the performance of our proposed reader module against different readers as described in Sec. V-B. Using predicted SFs in "SFReader" and "FFReader" hurts the performance compared to "UnfocusedReader" because the accuracy of the SFF module is not optimal. When we use gold SFs (optimal SFF), we see clear improvement over the "UnfocusedReader" baseline. When joint training "FFReader" with the SFF module, we see that it alleviates the inaccuracy of SFs. We hypothesize that the improvement comes from SFs being dynamic, and not treated as sub-optimal gold annotations. Joint training did not give any noticeable improvement when applied on the "UnfocusedReader," likely because it does not use any SFs, thus independent from the SFF module.

When joint training, even though the gradients pass through SFF and Reader modules separately, the amount of loss is what is impacted. When the Reader makes a wrong prediction based on wrong SFs, it is penalized more than if the predicted SFs were correct. This adjusts the Reader's dependence/confidence on the predicted SFs. In Table III, we see that if gold SFs are used, there is no need for joint training.

B. BET with SFF Module

In Table IV, we show the effect of BET on our SFF module. We also evaluate on gold paragraphs to see the upper bound of the SFs prediction, while unaffected by the accuracy of selecting correct paragraphs. The difference between gold and non gold evaluation is smaller with BET (2.58/1.91 vs 1.84/1.37), suggesting that the proposed BET not only improves the SFs

TABLE V
DEV SCORES OF JOINT *base* EXPERIMENTS ON THE DISTRACTOR SETTING OF HOTPOTQA WITH VARYING λ OF EQ. (5). ALL EXPERIMENTS USE BET.

Score of	Focus on Reader λ=0.1	0.25	0.5	0.75	Focus on SFF 0.9
	EM / F1	EM / F1	EM / F1	EM / F1	EM / F1
SFF module	61.92 / **88.55**	61.81 / 88.26	61.58 / 88.30	61.84 / 88.48	**62.07** / 88.52
FFReader module	65.04 / 78.62	64.90 / 78.54	**65.39** / **79.01**	64.59 / 78.57	64.50 / 78.20
Joint evaluation	**42.80** / 70.59	42.57 / 70.59	42.48 / **71.07**	42.39 / 70.72	42.62 / 70.47

TABLE VI
COMPARISON BETWEEN PAIR SCORING METHODS IN EVALUATION TIME USING THE *base* VERSION OF OUR FINAL MODEL (JOINTLY TRAINED FFREADER MODULE + SEPARATELY TRAINED SFF MODULE + BET).

Scoring method	Ans	Sup	Joint
	EM / F1	EM / F1	EM / F1
$-no\_ans^i_{logit}$	63.32 / 76.88	46.08 / 79.23	31.88 / 62.96
$\sum_{i \in S_i} P_i$	64.55 / 78.23	61.04 / 87.16	42.42 / 69.81
$\sum_{i \in S_i} P_i - no\_ans^i_{logit}$	**64.80** / **78.46**	**61.19** / **87.39**	**42.48** / **70.08**

TABLE VII
COMPARING JOINT FINE-TUNING WITH FRESH JOINT TRAINING AND SEPARATE TRAINING. WE USE BET WITH THE *base* VERSION OF THE MODEL, AND WE EVALUATE ONLY ON GOLD PARAGRAPHS.

Score of	Separate training	Fresh joint training	Joint fine-tuning
	EM / F1	EM / F1	EM / F1
SFF	**62.86** / 88.52	61.58 / 88.30	62.53 / **88.56**
FFReader	62.57 / 76.40	**65.39** / **79.01**	62.94 / 76.77
Joint	- / -	**42.48** / **71.07**	42.16 / 69.40

predictions, but also paragraph selection. We think BET is an important signal because of the way HotpotQA examples are collected; "Bridge" type questions, which are the majority, are anchored around the bridge entity connecting the paragraphs.

C. Effect of Joint Training on SFF Module

We see that joint training slightly harms the performance of the SFF module if BET is used, while improves it without BET. We think that it might be because even when a bridge entity is present in the training instances, there is no guarantee that there is an answer (the instances other than 2 gold paragraphs as explained in Sec. III-B), which may give conflicting signals for the BET existence. Even with this negative effect, the benefit of using BET outweighs the benefit of joint training the SFF module without BET.

D. Performance Per Question Type

We separate the results in Tables III and IV by question type, and we indeed see that BET benefits the "Bridge" question types much more than "Comparison" types. In some cases, it slightly harms "Comparison" types, likely because there is usually no hop between comparison paragraphs, so bridge entities act as distractions. As one possible future work, we can add a classifier to predict the question type and only tag bridge entities when the question is not a "Comparison" type.

Compared to the jointly trained FFReader, we see from Table III that there is almost no difference in score of "Comparison" question types when using gold SFs. This suggests that the flexible focusing on SFs is most important in "Bridge" question types.

E. Paragraph Pair Scoring

In addition to the method in Eq. (4), we experiment with two other alternative paragraph pair scoring methods as follows:

$$Pair_i = \sum_{j \in S_i} P_j \tag{6}$$

$$Pair_i = -no\_ans^i_{logit} \tag{7}$$

where we either only use paragraph classification scores from the SFF module, or we only use the no_answer score from the reader module. We show a comparison between the three methods in Table VI. We notice that the paragraph classification score in the SFF module is more important than no_ans_{logit} from the reader module, but their combination gives the best paragraph pair classification accuracy.

F. Initialization of Joint Training Experiments

All joint training experiments presented in this paper are *fresh* runs, meaning that the parameters of SFF and reader modules were initialized randomly. We also experiment with another type of training where those two modules were initialized with weights of separately trained SFF and reader modules. We fine-tune the initialized modules for less epochs and several learning rates, but we find that such training barely improves the reader. We compare the best performing trial against separate and fresh joint training in Table VII.

As mentioned in Sec. VII-A, FFReader suffers without joint training because it is trained with predicted SFs which are sub-optimal (Compared to the results in Table III where it is trained with gold SFs). Now fresh joint training alleviates this by using dynamic SFs and also the loss of the SFs misprediction helps the reader module become less dependent on the SFs when necessary (when they are wrong). The reader module in joint fine-tuning is initialized by a reader that was trained with sub-optimal SFs, and it only has limited training to fix its confidence in the sub-optimal SFs. We see that in the small improvement in Table VII where reader performance only improves about 0.35 EM/F1 scores.

G. Joint Training Hyperparameter λ

We experiment with different λ values in Eq. (5) and we show the details in Table V. We see that for all values, SFF performs worse than the separately trained SFF module, thus we opt for the λ value that most improves our reader module, which is 0.5.

TABLE VIII
LENGTH STATISTICS ABOUT THE INPUT OF DIFFERENT DATA SPLITS. TRAINING WITH NEG. MEANS ADDING NEGATIVE SAMPLES OF PARAGRAPH PAIRS TO THE 2 GOLD PARAGRAPH PAIR AS DESCRIBED IN SEC. III-B. WE USE THIS SPLIT IN OUR ACTUAL TRAINING.

Data split	Examples Count	$>$ 512	$>$ 768	$>$ 1,024
Training with neg.	270,817	19,693	1,830	371
Development	7,405	164	5	0

TABLE IX
A COMPARISON BETWEEN USING OUR MODEL WITH PLAIN ROBERTA VERSUS LONGROBERTA.

Score of	Plain RoBERTa	LongRoBERTa
	EM / F1	EM / F1
SFF	**62.5 / 88.89**	61.58 / 88.30
FFReader	64.32 / 78.31	**65.39 / 79.01**
Joint	42.80 / 70.85	**42.48 / 71.07**

TABLE X
EXAMPLE FROM HOTPOTQA WHERE SFS ANNOTATIONS ARE INCONCLUSIVE.

Question:	Which tennis player won more Grand Slam titles, Henri Leconte or Jonathan Stark?
Answer:	Jonathan Stark
Paragraph:	Jonathan Stark
[Gold SF] S_{1_1}	Jonathan Stark (born April 3, 1971) is a former professional tennis player from the United States.
[Gold SF] S_{1_2}	During his career he won two Grand Slam doubles titles (the 1994 French Open Men's Doubles and the 1995 Wimbledon Championships Mixed Doubles).
Paragraph:	Henri Leconte
S_{2_1}	Henri Leconte (born 4 July 1963) is a former French professional tennis player.
[Gold SF] S_{2_2}	He reached the men's singles final at the French Open in 1988, won the French Open men's doubles title in 1984, and helped France win the Davis Cup in 1991.

H. Plain RoBERTa versus LongRoBERTa

To justify the use of 1,024 tokens instead of 512 of a plain RoBERTa, we show statistics of input lengths in Table VIII. We see that 7% of our training instances (Training with neg.) and 2.2% of gold paragraph pairs in the dev split go over the 512 limit. To further study the effect of truncating those examples, we train our model using a plain RoBERTa instead of a LongRoBERTa as the basic encoding unit and show the results in Table IX. We find that the performance drops about 1.0/0.7 EM/F1 for the reader module which shows the benefit of using longer sequences. In Table II, models that outperform our model while only using 512 tokens also use GNNs, which explains their performance boost.

I. Case Study

In Table XI, we show some examples that are improved by our FFReader and BET proposals. We show links that are tagged with [BE][/BE] as underlined, wrong answers in {brackets}, correct answers in **bold** and supporting facts in *italic*.

Questions 1 and 2 demonstrate the effectiveness of our FFReader. In Question 1, Unfocused Reader tries to find a time span that can be the answer to the question from both paragraphs with no guidance on what sentences are important, while FFReader used the tagged SFs and showed how focusing on SFs helps finding the answer. Question 2 shows how our model can still predict answers outside the predicted SFs, while SFReader models like TAP2 and QUARK are limited to answers within the SFs. In this example, both SFReader and FFReader have the same predicted SFs, but SFReader can only see the SFs so the only driver name it can give is "Sergio Pérez," while FFReader considers the SFs but chooses the correct answer outside of them.

Questions 3 and 4 demonstrate the effect of our BET proposal. In question 3, we see that the Unfocused Reader without BET gives a correct answer type (a language), but the wrong answer. Without using BET, the system did not consider the importance of the second paragraph, and it just guessed one language near the word "Padosan." However, with the bridge clearly marking the importance of the linked document, the system was able to find the correct answer.In Question 4, FFReader without BET chose "Tunisian" as the answer because it appeared before "historian" that matches the question, without considering the other paragraph. With BET, the system paid more attention to the related paragraph and found the correct answer.

J. Problems in HotpotQA Annotations

In this section, we show that another reason why robustness in dealing with SFs is important is because annotations in HotpotQA can sometimes be inconclusive; they do not actually cover all the required sentences for reasoning. In Table X, we show an example of HotpotQA SFs annotation issue that we think hurts the training of the SFF module. We see that the sentence S_{2_1} is not considered as a gold SF, even though it includes the name of one of the entities in the question. If the meaning of "supporting facts" is that they are the only sentences required for reasoning to arrive at the answer, then if the reader has access *only* to these sentences, there is no way to resolve the pronoun "He" in sentence S_{2_1}. This would be considered an annotation mistake, and we encountered many such annotations.

If the meaning of "supporting facts" is that they are the core sentences required for making the final reasoning decision (not necessarily including all pronoun resolutions), then there is a logical annotation mistake, because the sentence that contains the answer is always a supporting fact, while other sentences that could have been the answer are not. An example on why this hurts the performance is as follows: S_{1_1} and S_{2_1} are equally important, they just define the players. If our SFF module predicts S_{2_1} as a SF, it would get penalized, even though this is a totally logical prediction. In order to have the SFF module achieve 100% EM accuracy, it would need to know the answer before predicting the SFs. After sampling 20 examples, we found 5 examples with this problem, which means around 25% of questions have this issue.

VIII. CONCLUSION

In this paper, we propose a multi-hop QA model that: 1) Uses supporting facts to answer questions in a novel way,

TABLE XI

EXAMPLES FROM THE DEVELOPMENT SPLIT OF HOTPOTQA DISTRACTOR SETTING. WE COMPARE THE RESULTS OF SEVERAL SYSTEMS THAT ARE SHOWN IN TABLE III AS FOLLOWS: QUESTION 1) UNFOCUSEDREADER V.S. FFREADER. QUESTION 2) SFREADER V.S. FFREADER. QUESTION 3) UNFOCUSEDREADER WITH AND WITHOUT BET. QUESTION 4) FFREADER WITH AND WITHOUT BET. ALL FFREADER MODELS WERE TRAINED WITH JOINT TRAINING WHILE USING A SEPARATELY TRAINED SFF MODULE, AS EXPLAINED IN SEC. V-D.

Question 1:	When was the Western Germanic language spoken from which the small settlement situated on the river Leda opposite Leer derives its name?
Gold Answer:	between the 8th and 16th centuries
Paragraph:	Leda (river)
	[SF] The Leda is a river in north-western Germany in the state of Lower Saxony. [/SF]
	It is a right tributary of the Ems and originates at the confluence of the Sagter Ems and the Soeste (Dreyschloot) near the town of Barßel.
	The Leda flows into the Ems near the town of Leer.
[Gold SF]	*[SF] On the southern bank of the Leda, in the "Overledingen Land" (Overledingen="country over the Leda"), opposite Leer, lies the small settlement of Kloster Muhde ("Muhde" from the Old Frisian "mutha" meaning "(river) mouth") [/SF].*
	The total length of the river is 29 km , of which the lower 1.9 km are navigable for sea-going vessels.
Paragraph:	Old Frisian
[Gold SF]	*[SF] Old Frisian is a West Germanic language spoken **between the 8th and 16th centuries** in the area between the Rhine and Weser on the European North Sea coast. [/SF]*
	The Frisian settlers on the coast of South Jutland (today's Northern Friesland) also spoke Old Frisian but no medieval texts of this area are known.
	The language of the earlier inhabitants of the region between the Zuiderzee and Ems River (the Frisians mentioned by Tacitus) is attested in only a few personal names and place-names.
	Old Frisian evolved into Middle Frisian, spoken from the {16th to the 19th century}.
Answer of UnfocusedReader with BET: 16th to the 19th century ([SF] tags were not used in UnfocusedReader)	
Answer of FFReader with BET: **between the 8th and 16th centuries**	
Question 2:	Which other Mexican formula one race car driver has held the podium besides the Force India driver born in 1990?
Gold Answer:	Pedro Rodríguez
Paragraph:	Forumula One drivers from Mexico
	There have been six Formula One drivers from Mexico who have taken part in races since the championship began in 1950.
[Gold SF]	**Pedro Rodríguez** is the most successful Mexican driver being the only one to have won a Grand Prix.
[Gold SF]	*[SF] Sergio Pérez, the only other Mexican to finish on the podium, currently races with Sahara Force India F1 team. [/SF]*
Paragraph:	Sergio Pérez
[Gold SF]	*[SF] {Sergio Pérez} Mendoza (; born 26 January 1990) also known as "Checo" Pérez, is a Mexican racing driver, currently driving for Force India. [/SF]*
Answer of SFReader with BET: Sergio Pérez	
Answer of FFReader with BET: **Pedro Rodríguez**	
Question 3:	Padosan had a supporting actor who is known as a successful playback singer in what language?
Gold Answer:	Hindi
Paragraph:	Padosan
	Padosan (Hindi: "", {English}: lady Neighbour) is a 1968 Indian comedy film.
	Directed by Jyoti Swaroop.
	It was produced by Mehmood, N. C. Sippy and written by Rajendra Krishan.
	It was a remake of the Bengali film "Pasher Bari" (1952) starring Bhanu Bandyopadhyay and Sabitri Chatterjee.
	The movie stars Sunil Dutt and Saira Banu in lead roles.
[Gold SF]	Kishore Kumar, Mukri, Raj Kishore and Keshto Mukherjee played the supporting roles.
	Mehmood as the South Indian musician and rival to Sunil Dutt is among the highlights of the film.
	It was considered as one of the best comedy movies ever made in Hindi film history.
	Mehmood's portrayal of a south Indian music teacher was one of his all time best and noted performances and a key highlight of the film.
	Kishore Kumar's character of a comical theater director was also well received.
	"Indiatimes Movies" ranked the movie amongst the "Top 25 Must See Bollywood Films".
	Music was composed by R.D. Burman and was a huge hit.
	Kishore Kumar sang for himself while Manna Dey sang for Mehmood.
Paragraph:	Kishore Kumar
	Kishore Kumar (4 August 1929 – 13 October 1987) was an Indian playback singer, actor, lyricist, composer, producer, director, and screenwriter.
[Gold SF]	He is considered one of the successful playback singers in the **Hindi** film industry.
Answer of UnfocusedReader without BET: English	
Answer of UnfocusedReader with BET: **Hindi**	
Question 4:	Georges-Henri Bousquet translated the work of a historian who is of what heritage?
Gold Answer:	North African Arab
Paragraph:	Georges-Henri Bousquet
	Georges-Henri Bousquet (21 June 1900, Meudon – 23 January 1978, Latresne) was a 20th-century French jurist, economist and Islamologist.
	He was Professor of law at the Faculty of Law of the University of Algiers where he was a specialist in the sociology of North Africa (Berbers, Islam).
[Gold SF]	He is also known for his translation work of the great Muslim authors, Al-Ghazali, a theologian who died in 1111 and {Tunisian} historian Ibn Khaldun (1332-1406).
	He was known as a polyglot, spoke several European languages (Dutch, his second mother tongue, English, German, Italian, but also Spanish, Danish, Norwegian ...) and Eastern ones (Arab, Malay ...).
Paragraph:	Ibn Khaldun
[Gold SF]	Ibn Khaldun (; Arabic: , "Abū Zayd 'Abd ar-Rahman ibn Muhammad ibn Khaldūn al-Hadrami" ; 27 May 1332 – 17 March 1406) was a **North African Arab** historiographer and historian.
Answer of FFReader without BET: Tunisian	
Answer of FFReader with BET: **North African Arab**	

2) Tags bridge entities that connect paragraph pairs, and 3) Jointly train separate modules for answer, and supporting facts prediction. Our model outperforms all non graph-based models in answer finding, and achieves comparable scores with state-of-the-art graph-based models. For future work, we want to explore applying global attention to entities to explore if it can mimic the GNNs that are applied on entities.

REFERENCES

[1] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ questions for machine comprehension of text," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 2383–2392.

[2] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord, "Think you have solved question answering? try arc, the ai2 reasoning challenge," *arXiv preprint arXiv:1803.05457*, 2018.

[3] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. Cohen, R. Salakhutdinov, and C. D. Manning, "HotpotQA: A dataset for diverse, explainable multi-hop question answering," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 2369–2380.

[4] J. Welbl, P. Stenetorp, and S. Riedel, "Constructing datasets for multi-hop reading comprehension across documents," *Transactions of the Association for Computational Linguistics*, vol. 6, pp. 287–302, 2018.

[5] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[6] Y. Fang, S. Sun, Z. Gan, R. Pillai, S. Wang, and J. Liu, "Hierarchical graph network for multi-hop question answering," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 8823–8838.

[7] M. Tu, K. Huang, G. Wang, J. Huang, X. He, and B. Zhou, "Select, answer and explain: Interpretable multi-hop reading comprehension over multiple documents," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 2020, pp. 9073–9080.

[8] N. Shao, Y. Cui, T. Liu, S. Wang, and G. Hu, "Is Graph Structure Necessary for Multi-hop Question Answering?" in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 7187–7192.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186.

[10] M. R. Glass, A. Gliozzo, R. Chakravarti, A. Ferritto, L. Pan, G. P. S. Bhargav, D. Garg, and A. Sil, "Span selection pre-training for question answering," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, Eds. Association for Computational Linguistics, 2020, pp. 2773–2782.

[11] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," *arXiv preprint arXiv:2004.05150*, 2020.

[12] A. Asai, K. Hashimoto, H. Hajishirzi, R. Socher, and C. Xiong, "Learning to retrieve reasoning paths over wikipedia graph for question answering," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[13] L. Qiu, Y. Xiao, Y. Qu, H. Zhou, L. Li, W. Zhang, and Y. Yu, "Dynamically fused graph network for multi-hop reasoning," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 6140–6150.

[14] D. Groeneveld, T. Khot, Mausam, and A. Sabharwal, "A simple yet strong pipeline for hotpotqa," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Association for Computational Linguistics, 2020, pp. 8839–8845.

[15] W. Chen, H. Zha, Z. Chen, W. Xiong, H. Wang, and W. Y. Wang, "Hybridqa: A dataset of multi-hop question answering over tabular and textual data," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16-20 November 2020*, T. Cohn, Y. He, and Y. Liu, Eds. Association for Computational Linguistics, 2020, pp. 1026–1036.

[16] J. Berant, A. Chou, R. Frostig, and P. Liang, "Semantic parsing on Freebase from question-answer pairs," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1533–1544.

[17] A. Bordes, N. Usunier, S. Chopra, and J. Weston, "Large-scale simple question answering with memory networks," *CoRR*, vol. abs/1506.02075, 2015.

[18] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, K. Toutanova, L. Jones, M. Kelcey, M.-W. Chang, A. M. Dai, J. Uszkoreit, Q. Le, and S. Petrov, "Natural questions: A benchmark for question answering research," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 452–466, Mar. 2019.

[19] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45.

[20] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[21] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[22] K. Clark, M. Luong, Q. V. Le, and C. D. Manning, "ELECTRA: pre-training text encoders as discriminators rather than generators," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[23] W.-t. Yih, M. Richardson, C. Meek, M.-W. Chang, and J. Suh, "The value of semantic parse labeling for knowledge base question answering," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 201–206. [Online]. Available: https://aclanthology.org/P16-2033

[24] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 1247–1250. [Online]. Available: https://doi.org/10.1145/1376616.1376746

[25] A. Talmor and J. Berant, "The web as a knowledge-base for answering complex questions," in *North American Association for Computational Linguistics (NAACL)*, 2018.

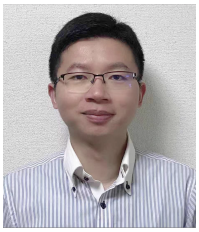
[26] G. He, Y. Lan, J. Jiang, W. X. Zhao, and J.-R. Wen, "Improving multi-hop knowledge base question answering by learning intermediate supervision signals," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, ser. WSDM '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 553–561. [Online]. Available: https://doi.org/10.1145/3437963.3441753

[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., 2017, pp. 5998–6008.

[28] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[29] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.
[30] M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontañón, P. Pham, A. Ravula, Q. Wang, L. Yang, and A. Ahmed, "Big bird: Transformers for longer sequences," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020.
[31] K. Nishida, K. Nishida, M. Nagata, A. Otsuka, I. Saito, H. Asano, and J. Tomita, "Answering while summarizing: Multi-task learning for multi-hop QA with evidence extraction," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 2335–2345.
[32] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.
[33] S. Maleki, M. Musuvathi, T. Mytkowicz, O. Saarikivi, T. Xu, V. Eksarevskiy, J. Ekanayake, and E. Barsoum, "Scaling distributed training with adaptive summation," *Proceedings of Machine Learning and Systems*, vol. 3, 2021.

Tareq Alkhaldi received his B.S. in Information Technology from Arab American University in 2010, and his M.S. in Informatics from Kyoto University in 2018. He is currently working towards a doctoral degree at Kyoto University. His research interests include natural language processing, and in particular, question answering and knowledge representation.



Chenhui Chu received his B.S. in software engineering from Chongqing University in 2008, and his M.S. and Ph.D. in Informatics from Kyoto University in 2012 and 2015, respectively. He is currently a program-specific associate professor at Kyoto University. His research interests include natural language processing, particularly machine translation and multimodal machine learning.

Sadao Kurohashi received the B.S., M.S., and Ph.D. in Electrical Engineering from Kyoto University in 1989, 1991 and 1994, respectively. He has been a visiting researcher of IRCS, University of Pennsylvania in 1994. He is currently a professor of the Graduate School of Informatics at Kyoto University. His research interests include natural language processing, knowledge acquisition/representation, and information retrieval. He received the 10th anniversary best paper award from the Journal of Natural Language Processing in 2004, 2009 Funai IT promotion award, and 2009 IBM faculty award.