Cognitive Graph for Multi-Hop Reading Comprehension at Scale

Ming Ding[†], **Chang Zhou**[‡], **Qibin Chen**[†], **Hongxia Yang**[‡], **Jie Tang**[†]

[†]Department of Computer Science and Technology, Tsinghua University
[‡]DAMO Academy, Alibaba Group
{dm18,chen-qb15}@mails.tsinghua.edu.cn
{ericzhou.zc,yang.yhx}@alibaba-inc.com
jietang@tsinghua.edu.cn

Abstract

We propose a new CogQA framework for multi-hop question answering in web-scale documents. Founded on the dual process theory in cognitive science, the framework gradually builds a *cognitive graph* in an iterative process by coordinating an **implicit extraction** module (System 1) and an **explicit reasoning** module (System 2). While giving accurate answers, our framework further provides explainable reasoning paths. Specifically, our implementation[1] based on BERT and graph neural network (GNN) efficiently handles millions of documents for multi-hop reasoning questions in the HotpotQA fullwiki dataset, achieving a winning joint F_1 score of 34.9 on the leaderboard, compared to 23.6 of the best competitor.[2]

1 Introduction

Deep learning models have made significant strides in machine reading comprehension and even outperformed human on single paragraph question answering (QA) benchmarks including SQuAD (Wang et al., 2018b; Devlin et al., 2018; Rajpurkar et al., 2016). However, to cross the chasm of reading comprehension ability between machine and human, three main challenges lie ahead: **1)** Reasoning ability. As revealed by adversarial tests (Jia and Liang, 2017), models for single paragraph QA tend to seek answers in sentences matched by the question, which does not involve complex reasoning. Therefore, multi-hop QA becomes the next frontier to conquer (Yang et al., 2018). **2)** Explainability. Explicit reasoning paths, which enable verification of logical rigor, are vital for the reliability of QA systems. HotpotQA (Yang et al., 2018) requires models to provide supporting sentences, which

---

[1]Codes: https://github.com/THUDM/CogQA
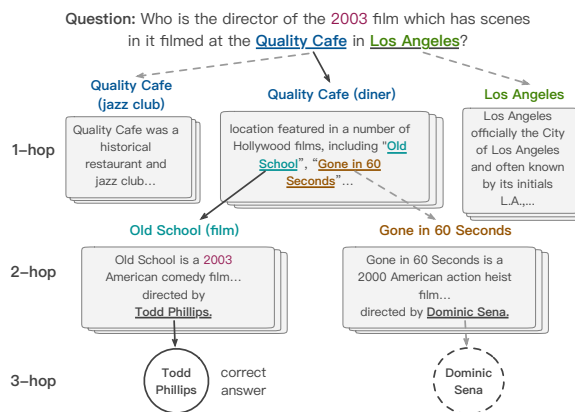[2]https://hotpotqa.github.io, March 4, 2019

Figure 1: An example of cognitive graph for multi-hop QA. Each *hop node* corresponds to an entity (e.g., "Los Angeles") followed by its introductory paragraph. The circles mean *ans nodes*, answer candidates to the question. Cognitive graph mimics human reasoning process. Edges are built when calling an entity to "mind". The solid black edges are the correct reasoning path.

means *unordered* and *sentence-level* explainability, yet humans can interpret answers with step by step solutions, indicating an *ordered* and *entity-level* explainability. **3)** Scalability. For any practically useful QA system, scalability is indispensable. Existing QA systems based on machine comprehension generally follow retrieval-extraction framework in DrQA (Chen et al., 2017), reducing the scope of sources to a few paragraphs by pre-retrieval. This framework is a simple compromise between single paragraph QA and scalable information retrieval, compared to human's ability to breeze through reasoning with knowledge in massive-capacity memory (Wang et al., 2003).

Therefore, insights on the solutions to these challenges can be drawn from the cognitive process of humans. *Dual process theory* (Evans, 1984, 2003, 2008; Sloman, 1996) suggests that our brains first retrieve relevant information following attention via an *implicit*, *unconscious* and *intu-*

itive process called **System 1**, based on which another *explicit*, *conscious* and *controllable* reasoning process, **System 2**, is then conducted. System 1 could provide resources according to requests, while System 2 enables diving deeper into relational information by performing sequential thinking in the working memory, which is slower but with human-unique rationality (Baddeley, 1992). For complex reasoning, the two systems are coordinated to perform *fast and slow thinking* (Kahneman and Egan, 2011) iteratively.

In this paper, we propose a framework, namely *Cognitive Graph QA* (CogQA), contributing to tackling all challenges above. Inspired by the dual process theory, the framework comprises functionally different System 1 and 2 modules. System 1 extracts question-relevant entities and answer candidates from paragraphs and encodes their semantic information. Extracted entities are organized as a *cognitive graph* (Figure 1), which resembles the working memory. System 2 then conducts the reasoning procedure over the graph, and collects *clues* to guide System 1 to better extract next-hop entities. The above process is iterated until all possible answers are found, and then the final answer is chosen based on reasoning results from System 2. An efficient implementation based on BERT (Devlin et al., 2018) and graph neural network (GNN) (Battaglia et al., 2018) is introduced.

Our contributions are as follows:

- We propose the novel CogQA framework for multi-hop reading comprehension QA at scale according to human cognition.
- We show that the *cognitive graph* structure in our framework offers ordered and entity-level explainability and suits for relational reasoning.
- Our implementation based on BERT and GNN surpasses previous works and other competitors substantially on all the metrics.

2 Cognitive Graph QA Framework

Reasoning ability of humankind depends critically on relational structures of information. Intuitively, we adopt a directed graph structure for step-by-step deduction and exploration in cognitive process of multi-hop QA. In our reading comprehension setting, each node in this *cognitive graph* $\mathcal{G}$ corresponds with an entity or possible answer x, also interchangeably denoted as node x. The extraction module System 1, reads the introductory

---

Algorithm 1: Cognitive Graph QA

Input:
System 1 model $\mathcal{S}_1$, System 2 model $\mathcal{S}_2$,
Question Q, Predictor $\mathcal{F}$, Wiki Database $\mathcal{W}$

1 Initialize cognitive graph $\mathcal{G}$ with entities mentioned in Q and mark them *frontier nodes*
2 **repeat**
3   pop a node x from frontier nodes
4   collect $clues[x, \mathcal{G}]$ from predecessor nodes of x
     // eg. $clues$ can be sentences where x is mentioned
5   fetch $para[x]$ in $\mathcal{W}$ if any
6   generate $sem[x, Q, clues]$ with $\mathcal{S}_1$ // initial $\mathbf{X}[x]$
7   **if** x *is a hop node* **then**
8   find hop and answer spans in $para[x]$ with $\mathcal{S}_1$
9   **for** y *in hop spans* **do**
10   **if** $y \notin \mathcal{G}$ *and* $y \in \mathcal{W}$ **then**
11   create a new hop node for y
12   **if** $y \in \mathcal{G}$ *and* $edge(x, y) \notin \mathcal{G}$ **then**
13   add edge (x, y) to $\mathcal{G}$
14   mark node y as a frontier node
15   **end**
16   **for** y *in answer spans* **do**
17   add new answer node y and edge (x, y) to $\mathcal{G}$
18   **end**
19   **end**
20   update hidden representation $\mathbf{X}$ with $\mathcal{S}_2$
21 **until** *there is no frontier node in $\mathcal{G}$ or $\mathcal{G}$ is large enough*;
22 **Return** $\arg\max_{\text{answer node } x} \mathcal{F}(\mathbf{X}[x])$

---

paragraph $para[x]$ of entity x and extracts *answer candidates* and useful *next-hop entities* from the paragraph. $\mathcal{G}$ is then expanded with these new nodes, providing explicit structure for the reasoning module, System 2. In this paper, we assume that System 2 conducts deep learning based instead of rule-based reasoning by computing hidden representations $\mathbf{X}$ of nodes. Thus System 1 is also required to summarize $para[x]$ into a semantic vector as initial hidden representation when extracting spans. Then System 2 updates $\mathbf{X}$ based on graph structure as reasoning results for downstream prediction.

Explainability is enjoyed owing to explicit reasoning paths in the cognitive graph. Besides simple paths, the cognitive graph can also clearly display joint or loopy reasoning processes, where new predecessors might bring new *clues* about the answer. *Clues* in our framework is a form-flexible concept, referring to information from predecessors for guiding System 1 to better extract spans. Apart from newly added nodes, those nodes with new incoming edges also need revisits due to new clues. We refer to both of them as *frontier nodes*.

Scalability means that the time consumption of QA will not grow significantly along with the number of paragraphs. Our framework can scale in nature since the only operation referred to all
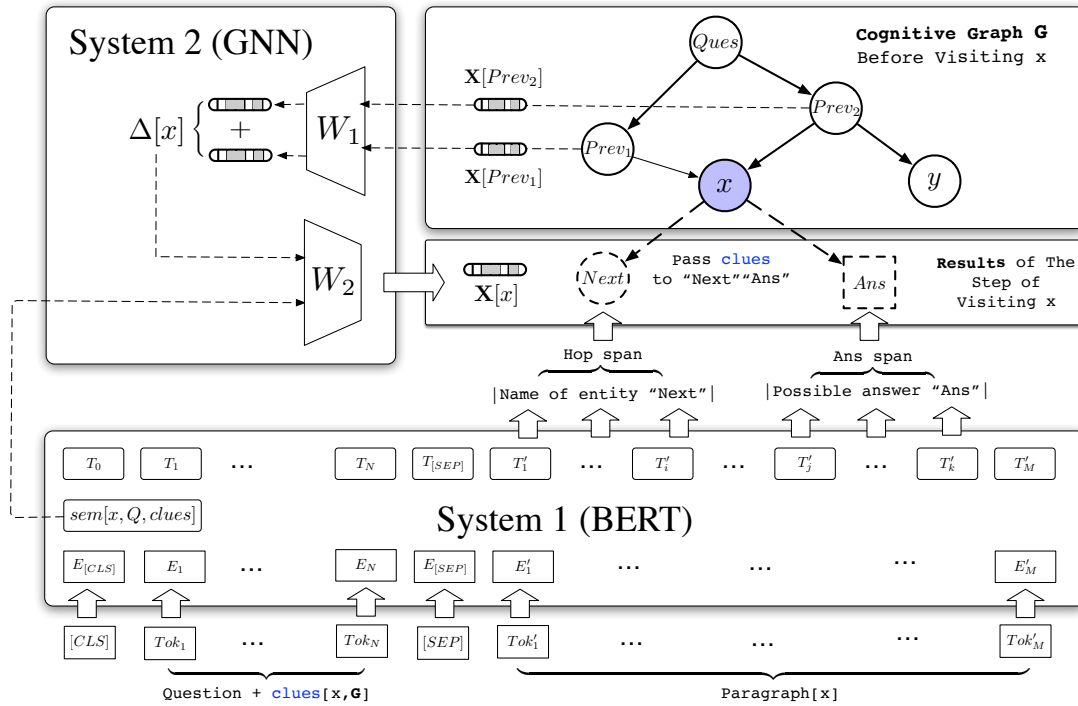
Figure 2: Overview of CogQA implementation. When visiting the node x, System 1 generates new hop and answer nodes based on the $clues[x, \mathcal{G}]$ discovered by System 2. It also creates the inital representation $sem[x, Q, clues]$, based on which the GNN in System 2 updates the hidden representations $\mathbf{X}[x]$.

paragraphs is to access some specific paragraphs by their title indexes. For multi-hop questions, traditional retrieval-extraction frameworks might sacrifice the potential of follow-up models, because paragraphs multiple hops away from the question could share few common words and little semantic relation with the question, leading to a failed retrieval. However, these paragraphs can be discovered by iteratively expanding with $clues$ in our framework.

Algorithm 1 describes the procedure of our framework CogQA. After initialization, an iterative process for graph expansion and reasoning begins. In each step we visit a frontier node x, and System 1 reads $para[x]$ under the guidance of $clues$ and the question Q, extracts spans and generates semantic vector $sem[x, Q, clues]$. Meanwhile, System 2 updates hidden representation $\mathbf{X}$ and prepares $clues[y, \mathcal{G}]$ for any successor node y. The final prediction is made based on $\mathbf{X}$.

3 Implementation

The main part to implement the CogQA framework is to determine the concrete models of System 1 and 2, and the form of $clues$.

Our implementation uses BERT as System 1 and GNN as System 2. Meanwhile, $clues[x, \mathcal{G}]$ are sentences in paragraphs of x's predecessor

nodes, from which x is extracted. We directly pass raw sentences as $clues$, rather than any form of computed hidden states, for easy training of System 1. Because raw sentences are self-contained and independent of computations from previous iterative steps, training at different iterative steps is then decoupled, leading to efficiency gains during training. Details are introduced in § 3.4. Hidden representations $\mathbf{X}$ for graph nodes are updated each time by a propagation step of GNN.

Our overall model is illustrated in Figure 2.

3.1 System 1

The extraction capacity of System 1 model is fundamental to construct the cognitive graph, thus a powerful model is needed. Recently, BERT (Devlin et al., 2018) has become one of the most successful language representation models on various NLP tasks, including SQuAD (Rajpurkar et al., 2016). BERT consists of multiple layers of Transformer (Vaswani et al., 2017), a self-attention based architecture, and is elaborately pre-trained on large corpora. Input sentences are composed of two different functional parts A and B.

We use BERT as System 1, and its input when visiting the node x is as follows:

$$\underbrace{[CLS] \; Question \; [SEP] \; clues[x, \mathcal{G}] \; [SEP]}_{Sentence \; A} \; \underbrace{Para[x]}_{Sentence \; B}$$

where $clues[x, \mathcal{G}]$ are sentences passed from predecessor nodes. The output vectors of BERT are denoted as $\mathbf{T} \in \mathbb{R}^{L \times H}$, where L is the length of the input sequence and H is the dimension size of the hidden representations.

It is worth noting that for answer node x, $Para[x]$ is probably missing. Thus we do not extract spans but can still calculate $sem[x, Q, clues]$ based on "Sentence A" part. And when extracting 1-hop nodes from question to initialize $\mathcal{G}$, we do not calculate semantic vectors and only the $Question$ part exists in the input.

Span Extraction Answers and next-hop entities have different properties. Answer extraction relies heavily on the character indicated by the question. For example "New York City" is more possible to be the answer of a *where* question than "2019", while next-hop entities are often the entities whose description matches statements in the question. Therefore, we predict answer spans and next-hop spans separately.

We introduce "pointer vectors" $\mathbf{S}_{hop}, \mathbf{E}_{hop}$, $\mathbf{S}_{ans}, \mathbf{E}_{ans}$ as additional learnable parameters to predict targeted spans. The probability of the ith input token to be the start of an answer span $P_{ans}^{start}[i]$ is calculated as follows:

$$P_{ans}^{start}[i] = \frac{e^{\mathbf{S}_{ans} \cdot \mathbf{T}_i}}{\sum_j e^{\mathbf{S}_{ans} \cdot \mathbf{T}_j}} \qquad (1)$$

Let $P_{ans}^{end}[i]$ be the probability of the ith input token to be the end of an answer span, which can be calculated following the same formula. We only focus on the positions with top K start probabilities $\{start_k\}$. For each k, the end position end_k is given by:

$$end_k = \underset{start_k \leq j \leq start_k + maxL}{\arg\max} P_{ans}^{end}[j] \qquad (2)$$

where $maxL$ is the maximum possible length of spans.

To identify irrelevant paragraphs, we leverage negative sampling introduced in § 3.4.1 to train System 1 to generate a *negative threshold*. In top K spans, those whose start probability is less than the negative threshold will be discarded. Because the 0th token $[CLS]$ is pre-trained to synthesize all input tokens for the Next Sentence Prediction task (Devlin et al., 2018), $P_{ans}^{start}[0]$ acts as the threshold in our implementation.

We expand the cognitive graph with remaining predicted answer spans as new "answer nodes". The same process is followed to expand "next-hop nodes" by replacing $\mathbf{S}_{ans}, \mathbf{E}_{ans}$ with $\mathbf{S}_{hop}, \mathbf{E}_{hop}$.

Semantics Generation As mentioned above, outputs of BERT at position 0 have the ability to summarize the sequence. Thus the most straightforward method is to use $\mathbf{T}_0$ as $sem[x, Q, clues]$. However, the last few layers in BERT are mainly in charge of transforming hidden representations for span predictions. In our experiment, the usage of the third-to-last layer output at position 0 as $sem[x, Q, clues]$ performs the best.

3.2 System 2

The first function of System 2 is to prepare $clues[x, \mathcal{G}]$ for frontier nodes, which we implement it as collecting the raw sentences of x's predecessor nodes that mention x.

The second function, to update hidden representations $\mathbf{X}$, is the core function of System 2. Hidden representations $\mathbf{X} \in \mathbb{R}^{n \times H}$ stand for the understandings of all n entities in $\mathcal{G}$. To fully understand the relation between an entity x and the question Q, barely analyzing semantics $sem[x, Q, clues]$ is insufficient. GNN has been proposed to perform deep learning on graph (Kipf and Welling, 2017), especially relational reasoning owing to the inductive bias of graph structure (Battaglia et al., 2018).

In our implementation, a variant of GNN is designed to serve as System 2. For each node x, the initial hidden representation $\mathbf{X}[x] \in \mathbb{R}^H$ is the semantic vector $sem[x, Q, clues]$ from System 1. Let $\mathbf{X}'$ be the new hidden representations after a propagation step of GNN, and $\Delta \in \mathbb{R}^{n \times H}$ be aggregated vectors passed from neighbours in the propagation. The updating formulas of $\mathbf{X}$ are as follows:

$$\Delta = \sigma((AD^{-1})^T \sigma(\mathbf{X}W_1)) \qquad (3)$$
$$\mathbf{X}' = \sigma(\mathbf{X}W_2 + \Delta) \qquad (4)$$

where σ is the activation function and $W_1, W_2 \in \mathbb{R}^{H \times H}$ are weight matrices. A is the adjacent matrix of $\mathcal{G}$, which is column-normalized to AD^{-1} where $D_{jj} = \sum_i A_{ij}$. Transformed hidden vector $\sigma(\mathbf{X}W_1)$ is left multiplied by $(AD^{-1})^T$, which can be explained as a localized spectral filter by Defferrard et al. (2016).

In the iterative step of visiting frontier node x, its hidden representation $\mathbf{X}[x]$ is updated following Equation (3)(4). In experiments, we observe that this "asynchronous updating" shows no apparent difference in performance with updating $\mathbf{X}$ of all the nodes together by multiple steps after $\mathcal{G}$ is

finalized, which is more efficient and adopted in practice.

3.3 Predictor

The questions in HotpotQA dataset generally fall into three categories: *special* question, *alternative* question and *general* question, which are treated as three different downstream prediction tasks taking $\mathbf{X}$ as input. In the test set, they can also be easily categorized according to interrogative words.

Special question is the most common case, requesting to find spans such as locations, dates or entity names in paragraphs. We use a two-layer fully connected network (FCN) to serve as predictor $\mathcal{F}$:

$$answer = \arg\max_{\text{answer node } x} \mathcal{F}(\mathbf{X}[x]) \qquad (5)$$

Alternative and general question both aims to compare a certain property of entity x and y in HotpotQA, respectively answered with entity name and "yes or no". These questions are regarded as binary classification with input $\mathbf{X}[x] - \mathbf{X}[y]$ and solved by another two identical FCNs.

3.4 Training

Our model is trained under a supervised paradigm with negative sampling. In the training set, the next-hop and answer spans are pre-extracted in paragraphs. More exactly, for each $para[x]$ relevant to question Q, we have spans data

$$\mathcal{D}[x, Q] = \{(y_1, start_1, end_1), ..., (y_n, start_n, end_n)\}$$

where the span from $start_i$ to end_i in $para[x]$ is fuzzy matched with the name of an entity or answer y_i. See § 4.1 for detail.

3.4.1 Task #1: Span Extraction

The ground truths of $P_{ans}^{start}, P_{ans}^{end}, P_{hop}^{start}, P_{hop}^{end}$ are constructed based on $\mathcal{D}[x, Q]$. There is at most one answer span $(y, start, end)$ in every paragraph, thus $\mathbf{gt}_{ans}^{start}$ is an one-hot vector where $\mathbf{gt}_{ans}^{start}[start] = 1$. However, multiple different next-hop spans might appear in one paragraph, so that $\mathbf{gt}_{hop}^{start}[start_i] = 1/k$ where k is the number of next-hop spans.

For the sake of the ability to discriminate irrelevant paragraphs, irrelevant *negative hop nodes* are added to $\mathcal{G}$ in advance. As mentioned in § 3.1, the output of $[CLS]$, $\mathbf{T}_0$, is in charge of generating *negative threshold*. Therefore, P_{ans}^{start} for each negative hop node is the one-hot vector where $\mathbf{gt}_{ans}^{start}[0] = 1$.

Cross entropy loss is used to train the span extraction task in System 1. The losses for the end position and for the next-hop spans are defined in the same way as follows.

$$\mathcal{L}_{ans}^{start} = -\sum_i \mathbf{gt}_{ans}^{start}[i] \cdot \log P_{ans}^{start}[i] \qquad (6)$$

3.4.2 Task #2: Answer Node Prediction

To command the reasoning ability, our model must learn to identify the correct answer node from a cognitive graph. For each question in the training set, we construct a training sample for this task. Each training sample is a composition of the *gold-only graph*, which is the union of all correct reasoning paths, and negative nodes. Negative nodes include negative hop nodes used in Task #1 and two negative answer nodes. A negative answer node is constructed from a span extracted at random from a randomly chosen hop node.

For special question, we first compute the *final answer probabilities* for each node by performing *softmax* on the outputs of $\mathcal{F}$. Loss $\mathcal{L}$ is defined as cross entropy between the probabilities and one-hot vector of answer node ans.

$$\mathcal{L} = -\log\left(softmax\big(\mathcal{F}(\mathbf{X})\big)[ans]\right) \qquad (7)$$

Alternative and general questions are optimized by binary cross entropy in similar ways. The losses of this task not only are back-propagated to optimize predictors and System 2, but also fine-tune System 1 through semantic vectors $sem[x, Q, clues]$.

4 Experiment

4.1 Dataset

We use the full-wiki setting of HotpotQA to conduct our experiments. 112,779 questions are collected by crowdsourcing based on the first paragraphs in Wikipedia documents, 84% of which require multi-hop reasoning. The data are split into a training set (90,564 questions), a development set (7,405 questions) and a test set (7,405 questions). All questions in development and test sets are *hard multi-hop* cases.

In the training set, for each question, an answer and paragraphs of 2 *gold* (useful) entities are provided, with multiple *supporting facts*, sentences containing key information for reasoning, marked out. There are also 8 unhelpful *negative paragraphs* for training. During evaluation, only questions are offered and meanwhile supporting facts are required besides the answer.

To construct cognitive graphs for training, edges in gold-only cognitive graphs are inferred from supporting facts by fuzzy matching based on Levenshtein distance (Navarro, 2001). For each supporting fact in $para[x]$, if any gold entity or the answer, denoted as y, is fuzzy matched with a span in the supporting fact, edge (x, y) is added.

4.2 Experimental Details

We use pre-trained BERT-base model released by (Devlin et al., 2018) in System 1. The hidden size H is 768, unchanged in node vectors of GNN and predictors. All the activation functions in our model are *gelu* (Hendrycks and Gimpel, 2016). We train models on Task #1 for 1 epoch and then on Task #1 and #2 jointly for 1 epoch. Hyperparameters in training are as follows:

Model	Task	batch size	learning rate	weight decay
BERT	#1,#2	10	$10^{-4}, 4 \times 10^{-5}$	0.01
GNN	#2	graph	10^{-4}	0

BERT and GNN are optimized by two different Adam optimizers, where $\beta_1 = 0.9, \beta_2 = 0.999$. The predictors share the same optimizer as GNN. The learning rate for parameters in BERT warmup over the first 10% steps, and then linearly decays to zero.

To select out supporting facts, we just regard the sentences in the *clues* of any node in graph as supporting facts. In the initialization of $\mathcal{G}$, these 1-hop spans exist in the question and can also be detected by fuzzy matching with supporting facts in training set. The extracted 1-hop entities by our framework can improve the retrieval phase of other models (See § 4.3), which motivated us to separate out the extraction of 1-hop entities to another BERT-base model for the purpose of reuse in implementation.

4.3 Baselines

The first category is previous work or competitor:

- **Yang et al. (2018)** The strong baseline model proposed in the original HotpotQA paper (Yang et al., 2018). It follows the retrieval-extraction framework of DrQA (2017) and subsumes the advanced techniques in QA, such as self-attention, character-level model, bi-attention.
- **GRN**, **QFE**, **DecompRC**, **MultiQA** The other models on the leaderboard.[3]

- **BERT** State-of-art model on single-hop QA. BERT in original paper requires single-paragraph input and pre-trained BERT can barely handle paragraphs of at most 512 tokens, much fewer than the average length of concatenated paragraphs. We add relevant sentences from predecessor nodes in the cognitive graph to every paragraphs and report the answer span with maximum start probability in all paragraphs.
- **Yang et al. (2018)-IR** Yang et al. (2018) with **I**mproved **R**etrieval. Yang et al. (2018) uses traditional *inverted index filtering* strategy to retrieve relevant paragraphs. The effectiveness might be challenged due to its failures to find out entities mentioned in question sometimes. The main reason is that word-level matching in retrieval usually neglect language models, which indicates importance and POS of words. We improve the retrieval by adding 1-hop entities spotted in the question by our model, increasing the coverage of supporting facts from 56% to 72%.

Another category is for ablation study:

- **CogQA-onlyR** model initializes $\mathcal{G}$ with the same entities retrieved in Yang et al. (2018) as 1-hop entities, mainly for fair comparison.
- **CogQA-onlyQ** initializes $\mathcal{G}$ only with 1-hop entities extracted from question, free of retrieved paragraphs. Complete CogQA implementation uses both.
- **CogQA-sys1** only retains System 1 and lacks cascading reasoning in System 2.

4.4 Results

Following Yang et al. (2018), the evaluation of answer and supporting facts consists of two metrics: Exact Match (EM) and F_1 score. Joint EM is 1 only if answer string and supporting facts are both strictly correct. Joint precision and recall are the products of those of Ans and Sup, and then joint F_1 is calculated. All results of these metrics are averaged over the test set.[4] Experimental results show superiority of our method in multiple aspects:

Overall Performance Our CogQA outperforms all baselines on all metrics by a significant margin (See Table 1). The leap of performance mainly

---

[3]All these models are unpublished before this paper.

[4]Thus it is possible that overall F_1 is lower than both precision and recall.

	Model	Ans				Sup				Joint			
		EM	F_1	Prec	Rec	EM	F_1	Prec	Rec	EM	F_1	Prec	Rec
Dev	Yang et al. (2018)	23.9	32.9	34.9	33.9	5.1	40.9	47.2	40.8	2.5	17.2	20.4	17.8
	Yang et al. (2018)-IR	24.6	34.0	35.7	34.8	10.9	49.3	52.5	52.1	5.2	21.1	22.7	23.2
	BERT	22.7	31.6	33.4	31.9	6.5	42.4	54.6	38.7	3.1	17.8	24.3	16.2
	CogQA-sys1	33.6	45.0	47.6	45.4	**23.7**	58.3	**67.3**	56.2	12.3	32.5	39.0	31.8
	CogQA-onlyR	34.6	46.2	48.8	46.7	14.7	48.2	56.4	47.7	8.3	29.9	36.2	30.1
	CogQA-onlyQ	30.7	40.4	42.9	40.7	23.4	49.9	56.5	48.5	**12.4**	30.1	35.2	29.9
	CogQA	**37.6**	**49.4**	**52.2**	**49.9**	23.1	**58.5**	64.3	**59.7**	12.2	**35.3**	**40.3**	**36.5**
Test	Yang et al. (2018)	24.0	32.9	-	-	3.86	37.7	-	-	1.9	16.2	-	-
	QFE	28.7	38.1	-	-	14.2	44.4	-	-	8.7	23.1	-	-
	DecompRC	30.0	40.7	-	-	N/A	N/A	-	-	N/A	N/A	-	-
	MultiQA	30.7	40.2	-	-	N/A	N/A	-	-	N/A	N/A	-	-
	GRN	27.3	36.5	-	-	12.2	48.8	-	-	7.4	23.6	-	-
	CogQA	**37.1**	**48.9**	-	-	**22.8**	**57.7**	-	-	**12.4**	**34.9**	-	-

Table 1: Results on HotpotQA (fullwiki setting). The test set is not public. The maintainer of HotpotQA only offers EM and F_1 for every submission. N/A means the model cannot find supporting facts.

results from the superiority of the CogQA framework over traditional retrieval-extraction methods. Since paragraphs that are multi-hop away may share few common words literally or even little semantic relation with the question, retrieval-extraction framework fails to find the paragraphs that become related only after the reasoning clues connected to them are found. Our framework, however, gradually discovers relevant entities following clues.

Logical Rigor QA systems are often criticized to answer questions with shallow pattern matching, not based on reasoning. To evaluate logical rigor of QA, we use $\frac{JointEM}{AnsEM}$, the proportion of "joint correct answers" in correct answers. The joint correct answers are those deduced from all necessary and correct supporting facts. Thus, this proportion stands for logical rigor of reasoning. The proportion of our method is up to 33.4%, far outnumbering 7.9% of Yang et al. (2018) and 30.3% of QFE.
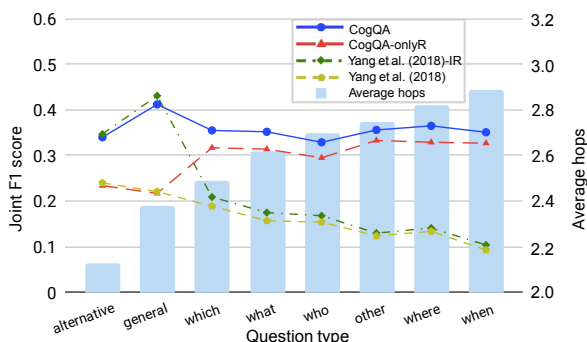

Figure 3: Model performance on 8 types of questions with different hops.

Multi-hop Reasoning Figure 3 illustrates joint F_1 scores and average hops of 8 types of questions, including general, alternative and special

questions with different interrogative word. As the hop number increases, the performance of Yang et al. (2018) and Yang et al. (2018)-IR drops dramatically, while our approach is surprisingly robust. However, there is no improvement in alternative and general questions, because the evidence for judgment cannot be inferred from supporting facts, leading to lack of supervision. Further human labeling is needed to answer these questions.

Ablation Studies To study the impacts of initial entities in cognitive graphs, CogQA-onlyR begins with the same initial paragraphs as (Yang et al., 2018). We find that CogQA-onlyR still performs significantly better. The performance decreases slightly compared to CogQA, indicating that the contribution mainly comes from the framework.

To compare against the retrieval-extraction framework, CogQA-onlyQ is designed that it only starts with the entities that appear in the question. Free of elaborate retrieval methods, this setting can be regarded as a natural thinking pattern of human being, in which only explicit and reliable relations are needed in reasoning. CogQA-onlyQ still outperforms all the baselines, which may reveal the superiority of CogQA framework over the retrieval-extraction framework.

BERT is not the key factor of improvement, although plays a necessary role. Vanilla BERT performs similar or even slightly poorer to (Yang et al., 2018) in this multi-hop QA task, possibly because of the pertinently designed architectures in Yang et al. (2018) to better leverage supervision of supporting facts.

To investigate the impacts of the absence of System 2, we design a System 1 only approach, CogQA-sys1, which inherits the iterative frame-
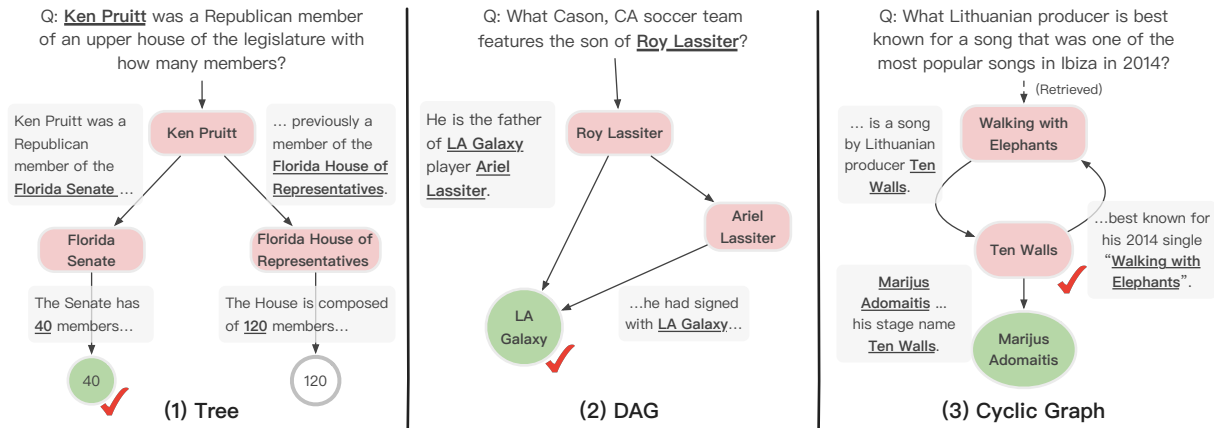
Figure 4: Case Study. Different forms of cognitive graphs in our results, i.e., Tree, Directed Acyclic Graph (DAG), Cyclic Graph. Circles are candidate answer nodes while rounded rectangles are hop nodes. Green circles are the final answers given by CogQA and check marks represent the annotated ground truth.

work but outputs answer spans with maximum predicted probability. On *Ans* metrics, the improvement over the best competitor decreases about 50%, highlighting the reasoning capacity of GNN on cognitive graphs.

Case Study We show how the cognitive graph clearly explains complex reasoning processes in our experiments in Figure 4. The cognitive graph highlights the heart of the question in case (1) – i.e., to choose between the number of members in two houses. CogQA makes the right choice based on semantic similarity between "Senate" and "upper house". Case (2) illustrates that the robustness of the answer can be boosted by exploring parallel reasoning paths. Case (3) is a *semantic retrieval* question without any entity mentioned, which is intractable for CogQA-onlyQ or even human. Once combined with information retrieval, our model finally gets the answer "Marijus Adomaitis" while the annotated ground truth is "Ten Walls". However, when backtracking the reasoning process in cognitive graph, we find that the model has already reached "Ten Walls" and answers with his real name, which is acceptable and even more accurate. Such explainable advantages are not enjoyed by black-box models.

5 Related work

Machine Reading Comprehension The research focus of machine reading comprehension (MRC) has been gradually transferred from cloze-style tasks (Hermann et al., 2015; Hill et al., 2015) to more complex QA tasks (Rajpurkar et al., 2016) recent years. Compared to the traditional computational linguistic pipeline (Hermann et al., 2015),

neural network models, for example BiDAF (Seo et al., 2017a) and R-net (Wang et al., 2017), exhibit outstanding capacity for answer extraction in text. Pre-trained on large corpra, recent BERT-based models nearly settle down the single paragraph MRC-QA problem with performances beyond human-level, driving researchers to pay more attention to multi-hop reasoning.

Multi-Hop QA Pioneering datasets of multi-hop QA are either based on limited knowledge base schemas (Talmor and Berant, 2018), or under multiple choices setting (Welbl et al., 2018). The noise in these datasets also restricted the development of multi-hop QA until high-quality HotpotQA (Yang et al., 2018) is released recently. The idea of "multi-step reasoning" also breeds *multi-turn* methods in single paragraph QA (Kumar et al., 2016; Seo et al., 2017b; Shen et al., 2017), assuming that models can capture information at deeper level implicitly by reading the text again.

Open-Domain QA Open-Domain QA (QA at scale) refers to the setting where the search space of the supporting evidence is extremely large. Approaches to get paragraph-level answers has been thoroughly investigated by the information retrieval community, which can be dated back to the 1990s (Belkin, 1993; Voorhees et al., 1999; Moldovan et al., 2000). Recently, DrQA (Chen et al., 2017) leverages a neural model to extract the accurate answer from retrieved paragraphs, usually called retrieval-extraction framework, greatly advancing this time-honored research topic again. Improvements are made to enhance retrieval by heuristic sampling (Clark and Gardner, 2018) or

reinforcement learning (Hu et al., 2018; Wang et al., 2018a), while for complex reasoning, necessary revisits to the framework are neglected.

6 Discussion and Conclusion

We present a new framework CogQA to tackle multi-hop machine reading problem at scale. The reasoning process is organized as cognitive graph, reaching unprecedented entity-level explainability. Our implementation based on BERT and GNN obtains state-of-art results on HotpotQA dataset, which shows the efficacy of our framework.

Multiple future research directions may be envisioned. Benefiting from the explicit structure in the cognitive graph, System 2 in CogQA has potential to leverage neural logic techniques to improve reliability. Moreover, we expect that prospective architectures combining attention and recurrent mechanisms will largely improve the capacity of System 1 by optimizing the interaction between systems. Finally, we believe that our framework can generalize to other cognitive tasks, such as conversational AI and sequential recommendation.

Acknowledgements

References

Alan Baddeley. 1992. Working memory. *Science*, 255(5044):556–559.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Nicholas J. Belkin. 1993. Interaction with texts: Information retrieval as information-seeking behavior. In *Information Retrieval*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1870–1879.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 845–855.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852.

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

Jonathan St BT Evans. 1984. Heuristic and analytic processes in reasoning. *British Journal of Psychology*, 75(4):451–468.

Jonathan St BT Evans. 2003. In two minds: dual-process accounts of reasoning. *Trends in cognitive sciences*, 7(10):454–459.

Jonathan St BT Evans. 2008. Dual-processing accounts of reasoning, judgment, and social cognition. *Annu. Rev. Psychol.*, 59:255–278.

Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *arXiv preprint arXiv:1606.08415*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301*.

Minghao Hu, Yuxing Peng, Zhen Huang, Xipeng Qiu, Furu Wei, and Ming Zhou. 2018. Reinforced mnemonic reader for machine reading comprehension. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4099–4106. AAAI Press.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031.

Daniel Kahneman and Patrick Egan. 2011. *Thinking, fast and slow*, volume 1. Farrar, Straus and Giroux New York.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*, pages 1378–1387.

Dan Moldovan, Sanda Harabagiu, Marius Pasca, Rada Mihalcea, Roxana Girju, Richard Goodrum, and Vasile Rus. 2000. The structure and performance of an open-domain question answering system. In *Proceedings of the 38th annual meeting on association for computational linguistics*, pages 563–570. Association for Computational Linguistics.

Gonzalo Navarro. 2001. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017a. Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations*.

Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2017b. Query-reduction networks for question answering. In *International Conference on Learning Representations*.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1047–1055. ACM.

Steven A Sloman. 1996. The empirical case for two systems of reasoning. *Psychological bulletin*, 119(1):3.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions.

In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 641–651.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Ellen M Voorhees et al. 1999. The trec-8 question answering track report. In *Trec*, volume 99, pages 77–82. Citeseer.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerry Tesauro, Bowen Zhou, and Jing Jiang. 2018a. r^3: Reinforced ranker-reader for open-domain question answering. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Wei Wang, Ming Yan, and Chen Wu. 2018b. Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1705–1714.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 189–198.

Yingxu Wang, Dong Liu, and Ying Wang. 2003. Discovering the capacity of human memory. *Brain and Mind*, 4(2):189–198.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association of Computational Linguistics*, 6:287–302.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.