Open Domain Question Answering (Part 1)

(Papers published between 2019~2020)

Department of Computer Science, Yonsei University

Seungone Kim

louisdebroglie@yonsei.ac.kr

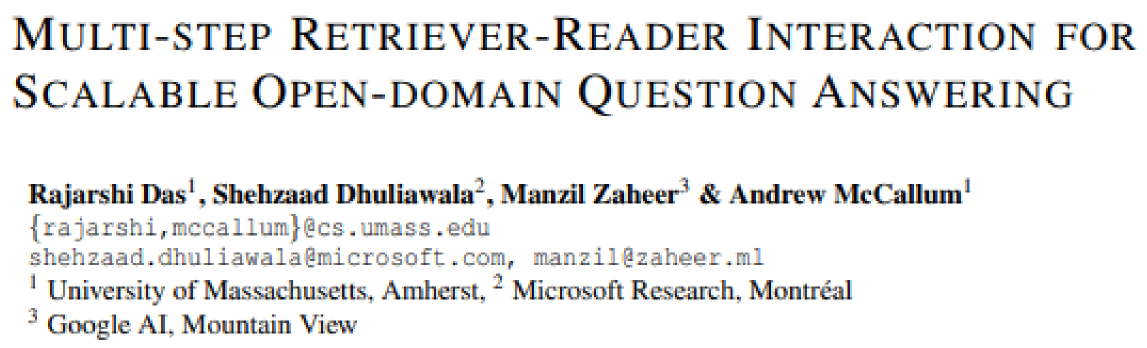
Referenced Papers

- Prerequisites / Additional Papers
    - **<Reading Comprehension>**
    - Dynamic coattention networks for question answering [ICLR 2017] : First introduced coattention scheme in QA
    - Bi-Directional Attention Flow for Machine Comprehension [ICLR 2017] : First introduced bi-directional attention
    - Simple and effective multi-paragraph reading comprehension [ACL 2018] : Integrate self-attention and bi-directional attention
    - Neural models for reasoning over multiple mentions using coreference [NAACL 2018] : Multi-hop reasoning
    - Coarse-grain fine-grain coattention network for multi-evidence question answering [ICLR 2019] : Multi-hop reasoning

    - **<Open-domain QA; Single Paragraph>**
    - Reading Wikipedia to answer open-domain questions [ACL 2017] : First introduced the use of neural methods to the task of Open-domain QA
    - R3 : Reinforced ranker-reader for open-domain question answering[AAAI 2018] : Used reinforcement learning to train ranker instead of TF-IDF
    - Denoising distantly supervised open-domain question answering [ACL 2018] : Argued the problem of relying on DS; propose a denoising method
    - Efficient and robust question answering from minimal context over documents [ACL 2018] : Trained sentence selector instead of document retrieval
    - Ranking paragraphs for improving answer recall in open-domain question answering [EMNLP 2018] : Precompute to encode questions and paragraphs
    - Evidence aggregation for answer re-ranking in open-domain question answering [ICLR 2018] : Precompute to encode questions and paragraphs
    - Phrase-indexed question answering : A new challenge for scalable document comprehension [EMNLP 2018] : Propose QA challenge to independently encode

    - Real-time open-domain question answering with dense-sparse phrase index [ACL 2019] : Proposed end-to-end models to jointly retrieve and read documents
    - Latent retrieval for weakly supervised open domain question answering [ACL 2019] : Proposed end-to-end models to jointly retrieve and read documents
    - Cognitive graph for multi-hop reading comprehension at scale [ACL 2019] : Incorporates entity links between documents and let the reader find the reasoning paths

- Key Papers
    - Multi-step retriever-reader interaction for scalable open-domain question answering [ICLR 2019]
    - Multi-hop paragraph retrieval for open-domain question answering [ACL 2019]
    - Learning to retrieve reasoning paths over Wikipedia graph for question answering [ICLR 2020]



Multi-step Retriever-Reader Interaction for Scalable Open-domain Question Answering

Rajarshi Das[1], Shehzaad Dhuliawala[2], Manzil Zaheer[3] & Andrew McCallum[1]
{rajarshi,mccallum}@cs.umass.edu
shehzaad.dhuliawala@microsoft.com, manzil@zaheer.ml
[1] University of Massachusetts, Amherst, [2] Microsoft Research, Montréal
[3] Google AI, Mountain View

ICLR 2019

Multi-Step Retriever-Reader Interaction For Scalable Open-Domain Question Answering [ICLR 2019]

- Performance of MRC systems degrades significantly when combined with a retriever in open domain setting
    - The EM accuracy of DrQA (Chen et al., 2017) on SQUAD dataset degrades from 69.5% to 28.4% in open-domain setting.
    - The primary reason for this degradation in performance is due to the **retriever's failure** to find the relevant paragraphs for the MRC model.

    - The authors propose two desiderata for general purpose open-domain QA systems.
    - <u>(a) The retriever model should be fast</u>
    - <u>(b) The retriever and reader models should be interactive.</u>

    - Previous Open-domain QA systems such as R3 and DS-QA have sophisticated retriever models where the reader and retriever are jointly trained.
    - However, their retriever computes **question-dependent paragraph representation** which is then encoded by running an expensive RNN over the tokens in the paragraph.
    - Since the retriever has a rank a lot of paragraphs, this design does not scale to large corporas.
    - On the other hand, the retriever model of DrQA is based on a TF-IDF retriever, but they **lack trainable parameters** and consequently unable to recover from mistakes.

Multi-Step Retriever-Reader Interaction For Scalable Open-Domain Question Answering [ICLR 2019]

- Model Architecture
  - The authors propose an architecture in which the retriever and reader iteratively interact with each other.
  - Also, the model **pre-computes and caches** representation of context, and hence can be computed and stored offline.
  - Given the input question, the retriever performs **Maximum Inner Product Search(MIPS)** with the **SGTree** Data Structure to find the most relevant contexts.
  - The highest ranked contexts are then passed to the neural MRC model.

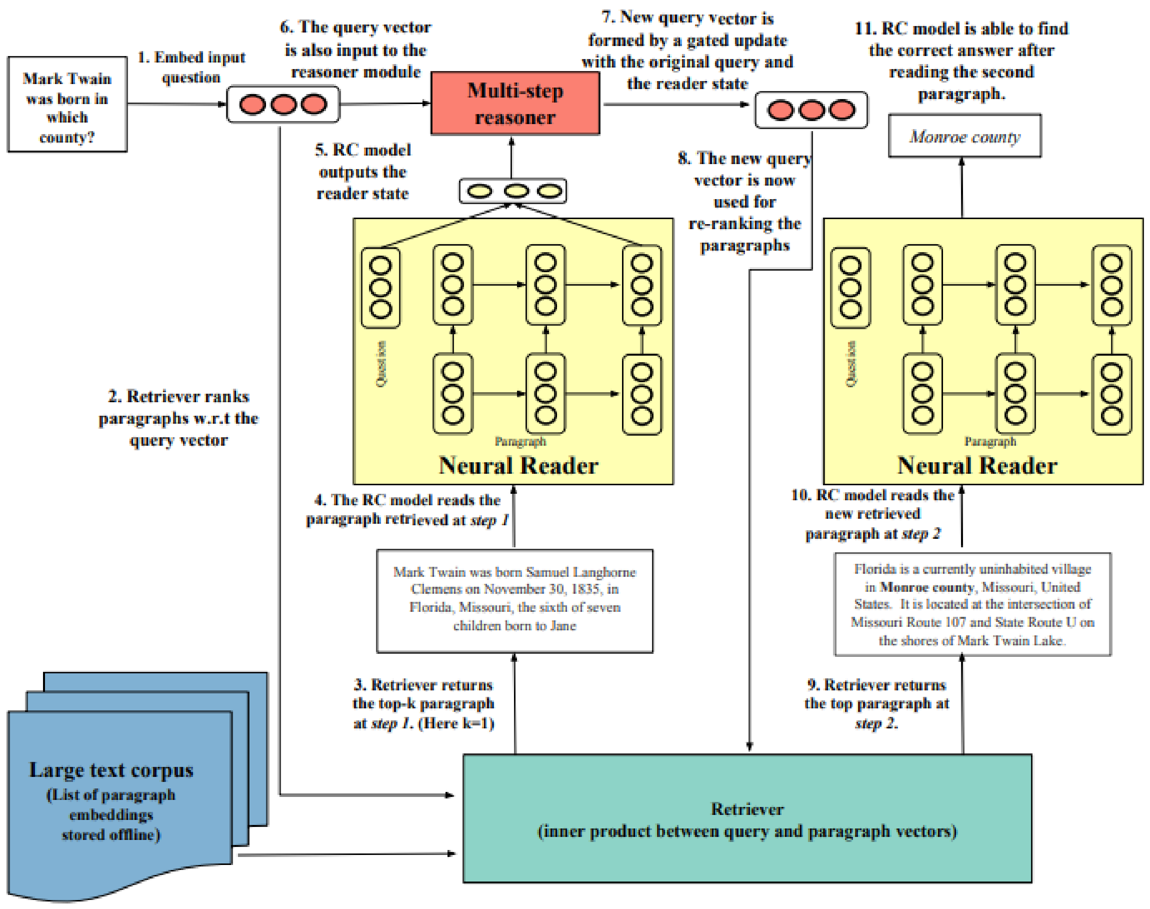
$$\begin{aligned}
\mathrm{NN}(\tilde{\mathbf{p}}_{\mathbf{i}}, \tilde{\mathbf{q}}) &= \arg\min_i \|\tilde{\mathbf{p}}_{\mathbf{i}} - \tilde{\mathbf{q}}\|^2 \\
&= \arg\min_i \|\tilde{\mathbf{p}}_{\mathbf{i}}\|^2 + \|\tilde{\mathbf{q}}\|^2 - 2\langle \tilde{\mathbf{p}}_{\mathbf{i}}, \tilde{\mathbf{q}} \rangle \\
&= \arg\min_i (\mathbf{u}^2 - 2\langle \tilde{\mathbf{p}}_{\mathbf{i}}, \tilde{\mathbf{q}} \rangle) \\
&= \arg\max_i \langle \tilde{\mathbf{p}}_{\mathbf{i}}, \tilde{\mathbf{q}} \rangle = \arg\max_i \langle \mathbf{p}_{\mathbf{i}}, \mathbf{q} \rangle = \mathrm{MIPS}(\mathbf{p}_{\mathbf{i}}, \mathbf{q})
\end{aligned}$$

  - Since the answer might not exist in the initial retrieved paragraphs or the model would need to combine information across multiple paragraphs, the reader is used to generate a **new query vector** that will re-rank the context.
  - This allows the model to read new paragraphs and combine evidence across multiple paragraphs.
  - Since the retriever makes a hard selection of paragraphs to send to the reader, the retriever and reader are jointly trained using **reinforcement learning**.

$$\log\left(\frac{\sum_{j\in\mathcal{P}}\sum_{k\in\mathcal{I}(w_s,p_j)}\exp(\mathrm{score}_{\mathrm{start}}(k,j))}{\sum_{j\in\mathcal{P}}\sum_{i=1}^{n_j}\exp(\mathrm{score}_{\mathrm{start}}(i,j))}\right)$$

Multi-Step Retriever-Reader Interaction For Scalable Open-Domain Question Answering [ICLR 2019]

- Model Architecture



Mark Twain was born in which county?
1. Embed input question
6. The query vector is also input to the reasoner module
Multi-step reasoner
7. New query vector is formed by a gated update with the original query and the reader state
11. RC model is able to find the correct answer after reading the second paragraph.
Monroe county
5. RC model outputs the reader state
8. The new query vector is now used for re-ranking the paragraphs
Question
Paragraph
Neural Reader
Question
Paragraph
Neural Reader
2. Retriever ranks paragraphs w.r.t the query vector
4. The RC model reads the paragraph retrieved at step 1
10. RC model reads the new retrieved paragraph at step 2
Mark Twain was born Samuel Langhorne Clemens on November 30, 1835, in Florida, Missouri, the sixth of seven children born to Jane
Florida is a currently uninhabited village in Monroe county, Missouri, United States. It is located at the intersection of Missouri Route 107 and State Route U on the shores of Mark Twain Lake.
3. Retriever returns the top-k paragraph at step 1. (Here k=1)
9. Retriever returns the top paragraph at step 2.
Large text corpus (List of paragraph embeddings stored offline)
Retriever (inner product between query and paragraph vectors)

Multi-Step Retriever-Reader Interaction For Scalable Open-Domain Question Answering [ICLR 2019]

- Multi-Step Reasoner
  - The multi-step reasoner, a module comprised of GRUs take in the current state of the reader and the current query vector and does a gated update to produce a **reformulated query**.
  - More formally, the multi-step reasoner is given the current query representation $q_t \in \mathbb{R}^{2d}$ and j-th hidden vector of the paragraph $m_j \in \mathbb{R}^{2p}$.
  - With pooling operation on the hidden representation of each question token, we acquire L.
  - The reader state is computed using m_j and L, which is then used with q_t to acquire q_{t+1}.

$$\alpha_{\mathbf{j}} = \frac{\exp\left(\mathbf{m_j} \cdot \mathbf{L}\right)}{\sum_{j'} \exp\left(\mathbf{m'_j} \cdot \mathbf{L}\right)} \qquad \mathbf{S} = \sum_j \left(\alpha_{\mathbf{j}} \cdot \mathbf{m_j}\right) \qquad \begin{aligned} \mathbf{q}'_{\mathbf{t+1}} &= \mathrm{GRU}\left(\mathbf{q_t}, \mathbf{S}\right) \\ \mathbf{q_{t+1}} &= \mathrm{FFN}(\mathbf{q}'_{\mathbf{t+1}}) \end{aligned}$$

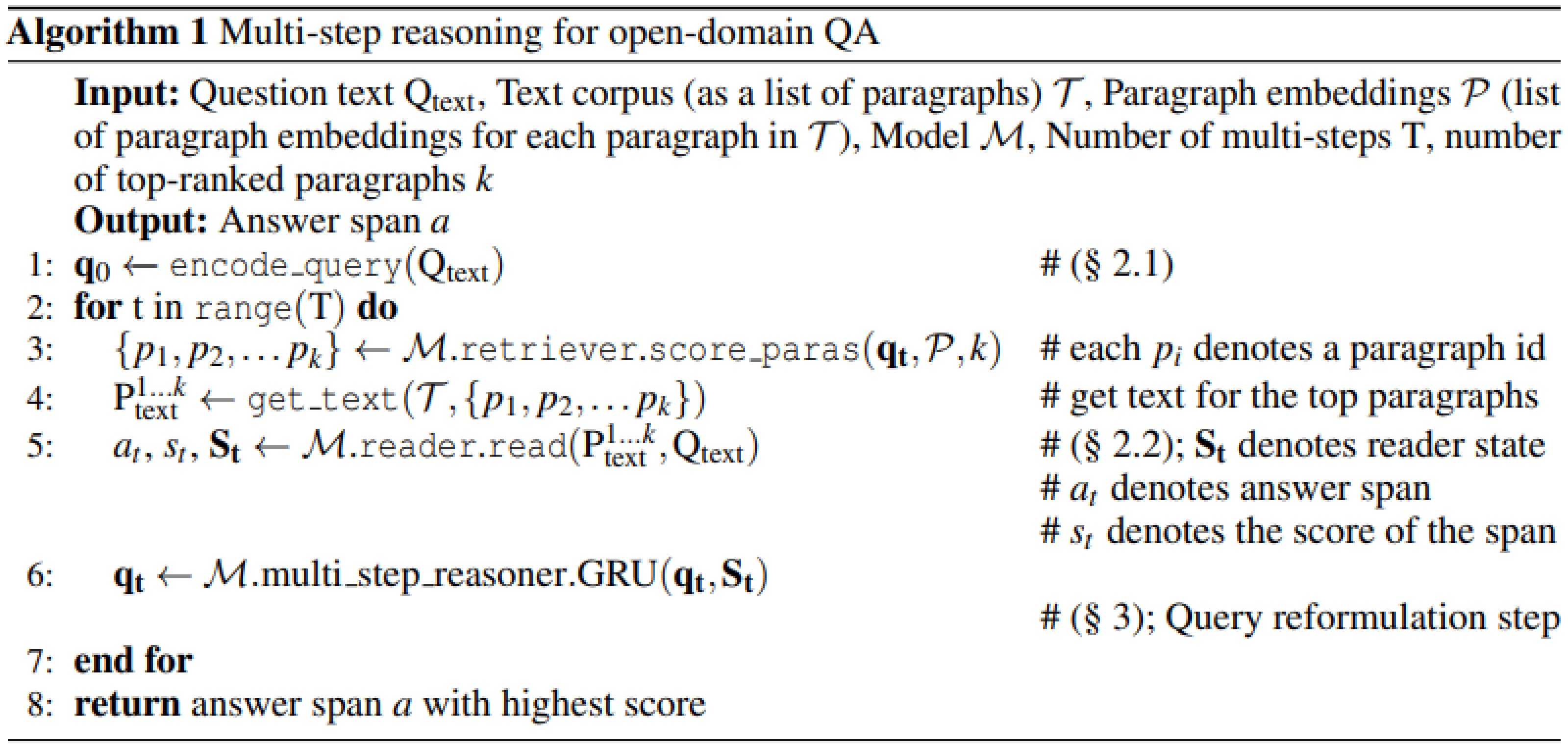
- Training with Reinforcement Learning
  - There exists no supervision for training the query reformulation of the multi-step reasoner, so the authors employ **reinforcement learning**.
  - The authors define the problem as a **deterministic finite horizon Partially Observed decision process**(POMDP).
  - The **policy π** is parameterized by GRU and the FFN of the multi-step reasoner, and the **reward(F1 score)** is returned by environment at time t.

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi \left[\sum_{t=1}^{T} r_t \cdot \log(\pi_\theta(p_t \mid q)) \right]$$

$$\pi_\theta(p_t \mid q) = \mathrm{softmax}(\mathrm{score}(\mathrm{p_t}, \mathrm{q}))$$

Multi-Step Retriever-Reader Interaction For Scalable Open-Domain Question Answering [ICLR 2019]

- Training

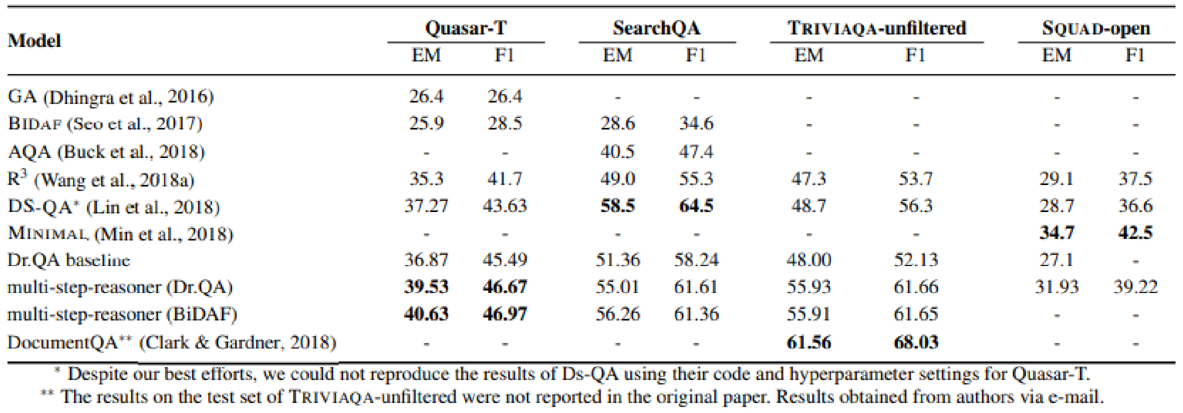


Algorithm 1 Multi-step reasoning for open-domain QA

```
    Input: Question text $Q_{\text{text}}$, Text corpus (as a list of paragraphs) $\mathcal{T}$, Paragraph embeddings $\mathcal{P}$ (list
    of paragraph embeddings for each paragraph in $\mathcal{T}$), Model $\mathcal{M}$, Number of multi-steps T, number
    of top-ranked paragraphs $k$
    Output: Answer span $a$
1:  $\mathbf{q_0} \leftarrow$ encode_query($Q_{\text{text}}$)                                    # (§ 2.1)
2:  for t in range(T) do
3:      $\{p_1, p_2, \dots p_k\} \leftarrow \mathcal{M}$.retriever.score_paras($\mathbf{q_t}, \mathcal{P}, k$)   # each $p_i$ denotes a paragraph id
4:      $P_{\text{text}}^{1\dots k} \leftarrow$ get_text($\mathcal{T}, \{p_1, p_2, \dots p_k\}$)        # get text for the top paragraphs
5:      $a_t, s_t, \mathbf{S_t} \leftarrow \mathcal{M}$.reader.read($P_{\text{text}}^{1\dots k}, Q_{\text{text}}$)      # (§ 2.2); $\mathbf{S_t}$ denotes reader state
                                                                       # $a_t$ denotes answer span
                                                                       # $s_t$ denotes the score of the span
6:      $\mathbf{q_t} \leftarrow \mathcal{M}$.multi_step_reasoner.GRU($\mathbf{q_t}, \mathbf{S_t}$)
                                                                       # (§ 3); Query reformulation step
7:  end for
8:  return answer span $a$ with highest score
```

Multi-Step Retriever-Reader Interaction For Scalable Open-Domain Question Answering [ICLR 2019]

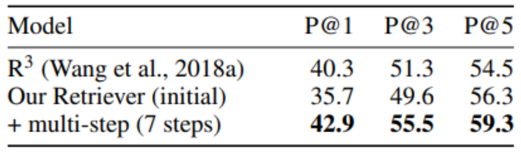
- Experiments / Results



Model	**Quasar-T**		**SearchQA**		**TRIVIAQA-unfiltered**		**SQUAD-open**	
	EM	F1	EM	F1	EM	F1	EM	F1
GA (Dhingra et al., 2016)	26.4	26.4	-	-	-	-	-	-
BIDAF (Seo et al., 2017)	25.9	28.5	28.6	34.6	-	-	-	-
AQA (Buck et al., 2018)	-	-	40.5	47.4	-	-	-	-
R^3 (Wang et al., 2018a)	35.3	41.7	49.0	55.3	47.3	53.7	29.1	37.5
DS-QA* (Lin et al., 2018)	37.27	43.63	**58.5**	**64.5**	48.7	56.3	28.7	36.6
MINIMAL (Min et al., 2018)	-	-	-	-	-	-	**34.7**	**42.5**
Dr.QA baseline	36.87	45.49	51.36	58.24	48.00	52.13	27.1	-
multi-step-reasoner (Dr.QA)	**39.53**	**46.67**	55.01	61.61	55.93	61.66	31.93	39.22
multi-step-reasoner (BiDAF)	**40.63**	**46.97**	56.26	61.36	55.91	61.65	-	-
DocumentQA** (Clark & Gardner, 2018)	-	-	-	-	**61.56**	**68.03**	-	-

* Despite our best efforts, we could not reproduce the results of Ds-QA using their code and hyperparameter settings for Quasar-T.
** The results on the test set of TRIVIAQA-unfiltered were not reported in the original paper. Results obtained from authors via e-mail.

Table 2: Performance on test sets for various datasets



Model	P@1	P@3	P@5
R^3 (Wang et al., 2018a)	40.3	51.3	54.5
Our Retriever (initial)	35.7	49.6	56.3
+ multi-step (7 steps)	**42.9**	**55.5**	**59.3**

Table 3: Retrieval performance on QUASAR-T. The match-LSTM based retriever of R^3 is a more powerful model than our intial retrieval model. However, after few steps of multi-step-reasoner, the performance increases suggesting that re-ranking via query-reformulation is retrieving relevant evidence from the corpus. We report the P@k on the last step.

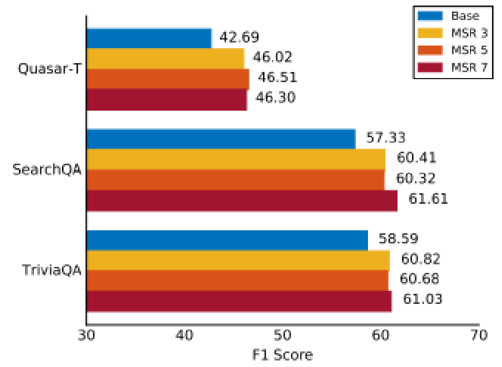


Figure 3: F1 score w.r.t number of steps.

Multi-Hop Paragraph Retrieval for Open-Domain Question Answering

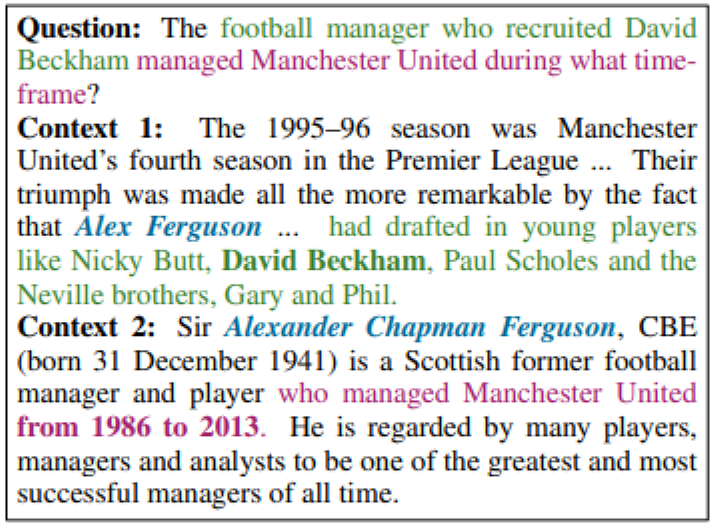
Yair Feldman and **Ran El-Yaniv**
Department of Computer Science
Technion – Israel Institute of Technology
Haifa, Israel
`{yairf11, rani}@cs.technion.ac.il`

ACL 2019

Multi-Hop Paragraph Retrieval for Open-Domain Question Answering [ACL 2019]

- This is the first work to propose a viable solution for **open-domain multi-hop QA**
  - Previously there has been several works that discuss solutions for multi-hop reasoning but not in the domain of open-domain multi-hop QA.
  - This task is particularly challenging since it requires the simultaneous performance of **textual reasoning** and **efficient searching**.

  - The authors present a method for retrieving multiple supporting paragraphs, nested amidst a large knowledge base, which contain the necessary evidence to answer a given question.
  - The key idea is to **iteratively retrieve** supporting paragraphs by forming a **joint vector representation** of both a question and a paragraph.



Question: The football manager who recruited David Beckham managed Manchester United during what time-frame?
Context 1: The 1995–96 season was Manchester United's fourth season in the Premier League ... Their triumph was made all the more remarkable by the fact that ***Alex Ferguson*** ... had drafted in young players like Nicky Butt, **David Beckham**, Paul Scholes and the Neville brothers, Gary and Phil.
Context 2: Sir ***Alexander Chapman Ferguson***, CBE (born 31 December 1941) is a Scottish former football manager and player who managed Manchester United **from 1986 to 2013**. He is regarded by many players, managers and analysts to be one of the greatest and most successful managers of all time.

Multi-Hop Paragraph Retrieval for Open-Domain Question Answering [ACL 2019]

- Task Definition
  - Define **open-domain QA task** by a triplet (KS, Q, A) where $KS = \{P_1, P_2, \dots, P_{|KS|}\}$ is a background knowledge source.
  - The objective of this task is to find the answer A to the question Q using background knowledge source KS.
  - Formally, the task is to learn a function ϕ such that $A = \phi(Q, KS)$.

  - In contrast to the single-hop case, in the **multi-hop retrieval scenario**,
    there are types of questions whose answers can be only be inferred by using at least two different paragraphs.
  - When the second context is observed alone, it may appear to be irrelevant to the question at hand
    and the information in the first context is necessary to retrieve the second part of the evidence correctly.

Multi-Hop Paragraph Retrieval for Open-Domain Question Answering [ACL 2019]

- Methodology (MUPPET : Multi-hoP Paragraph rETrieval)
  - The overall scheme consists of two main components : (a) **a paragraph and question encoder**, and (b) **a paragraph reader**.
  - The **encoder** is trained to encode paragraphs into d-dimensional vectors, and to encode questions into search vectors in the same vector space.
  - Then, a **MIPS(Maximum Inner Product Search) algorithm** is applied to find the most similar paragraphs to a given question.
  - The similar paragraphs are then passed to the **paragraph reader**, which, in turn, extracts the most probable answer to the question.

  - Given a question Q, its encoding $q \in \mathbb{R}^d$ is obtained using the encoder and is transformed into a search vector $q^s \in \mathbb{R}^d$.
  - With subsequent retrieval iteration, the top-k relevant paragraphs $\{P_1{}^Q, P_2{}^Q. \ldots, P_k{}^Q\} \subset KS$ are retrieved using MIPS.
  - Note that paragraphs retrieved in its previous iteration reformulate the search vector, which means that k new search vectors, $\{\widetilde{q_1}^s, \widetilde{q_2}^s, \ldots, \widetilde{q_k}^s\}$ are produced.
  - This method can be seen as performing beam search of width k in the encoded paragraphs' space.

Multi-Hop Paragraph Retrieval for Open-Domain Question Answering [ACL 2019]

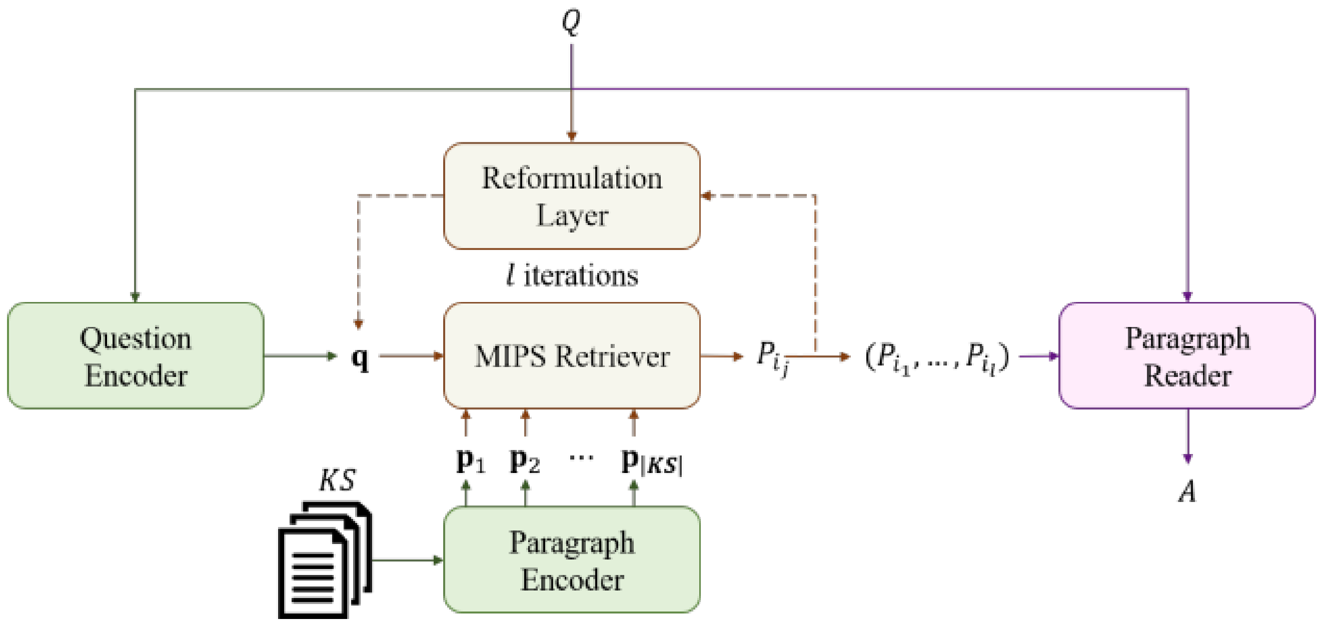
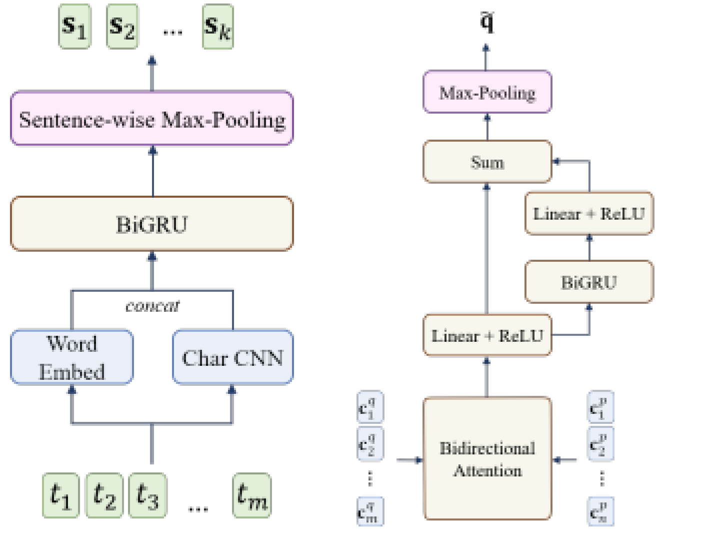


Figure 2: A high-level overview of our solution, MUPPET.



(a) Sentence Encoder (b) Reformulation Component

Figure 3: Architecture of the main components of our paragraph and question encoder. (a) Our sentence encoder architecture. The model receives a series of tokens as input and produces a sequence of sentence representations. (b) Our reformulation component architecture. This layer receives contextualized representations of a question and a paragraph, and produces a reformulated representation of the question.

Multi-Hop Paragraph Retrieval for Open-Domain Question Answering [ACL 2019]

- Relevance Scores / Search Vector Derivation
  - Given the sentence representations of paragraph P and the question encoding q, the **relevance score** of P w,r,t q is calculated.

$$rel(Q,P) = \max_{i=1,\dots,k} \sigma\left(\begin{bmatrix} \mathbf{s}_i \\ \mathbf{s}_i \odot \mathbf{q} \\ \mathbf{s}_i \cdot \mathbf{q} \\ \mathbf{q} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ w_3 \\ \mathbf{w}_4 \end{bmatrix} + b \right)$$

  - The **final search vector** to execute MIPS in the paragraph encoding space is executed at every iteration.

$$rel(Q,P) \propto \max_{i=1,\dots,k} \mathbf{s}_i^\top (\mathbf{w}_1 + \mathbf{w}_2 \odot \mathbf{q} + w_3 \cdot \mathbf{q}).$$

$$\mathbf{q}_s = \mathbf{w}_1 + \mathbf{w}_2 \odot \mathbf{q} + w_3 \cdot \mathbf{q}$$

  - The training objective for each iteration is composed of two components, a **binary cross-entropy loss function** and a **ranking loss function**.

$$\mathcal{L}_{CE} = -\frac{1}{N} \sum_{i=1}^{N} y_i \log\left(rel(Q_i, P_i)\right) + (1 - y_i) \log\left(1 - rel(Q_i, P_i)\right)$$

$$\mathcal{L}_R = \frac{1}{M} \sum_{i=1}^{M} \max(0, \gamma - q_i^{pos} + q_i^{neg})$$

$$\mathcal{L} = \mathcal{L}_{CE} + \lambda \mathcal{L}_R$$

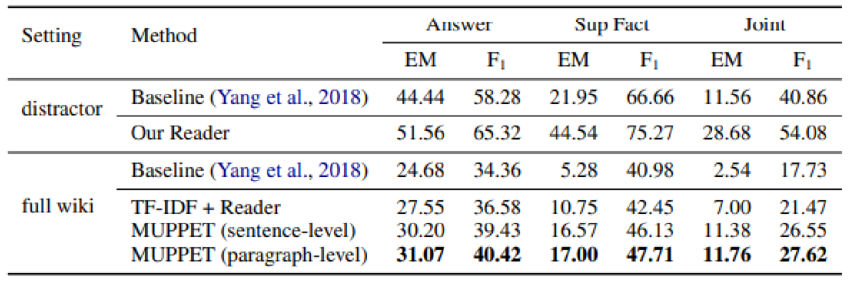
Multi-Hop Paragraph Retrieval for Open-Domain Question Answering [ACL 2019]

- Paragraph Reader
  - The paragraph reader receives as input a question Q and a paragraph P and extracts the most probable answer span to Q from P.
  - The authors use the S-norm model proposed by *Clark and Gardner(2018)*.
  - The paragraph reader optimize the negative log-likelihood function used in the S-norm model for the span start & end boundaries.

$$\mathcal{L}_{start} = -\log\left(\frac{\sum_{j\in P^Q}\sum_{k\in A_j} e^{s_{kj}}}{\sum_{j\in P^Q}\sum_{i=1}^{n_j} e^{s_{ij}}}\right) \qquad \mathcal{L}_{span} = \mathcal{L}_{start} + \mathcal{L}_{end}.$$

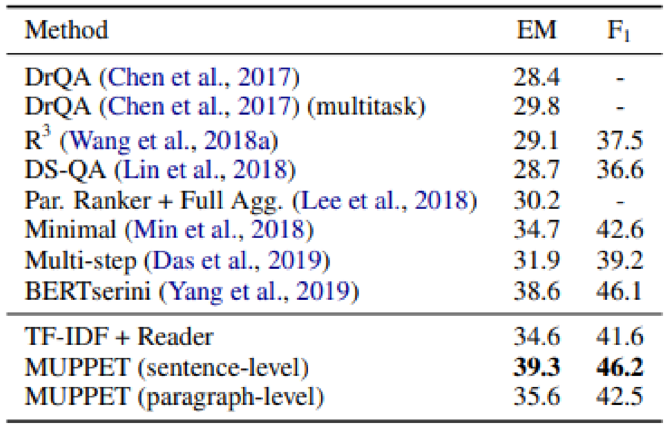
Multi-Hop Paragraph Retrieval for Open-Domain Question Answering [ACL 2019]

- Experiments
  - The authors use HotpotQA as a dataset to test multi-hop retrieval, and SQuAD-Open as a dataset to test single-hop retrieval.
  - The authors used a width of 8 in the first retrieval iteration, where the top 45 paragraphs are fed to the reader independently to find the most probable answer.
  - There is a **trade-off** for choosing the various values of width (***higher recall vs less noise***)



Setting	Method	Answer		Sup Fact		Joint	
		EM	F_1	EM	F_1	EM	F_1
distractor	Baseline (Yang et al., 2018)	44.44	58.28	21.95	66.66	11.56	40.86
	Our Reader	51.56	65.32	44.54	75.27	28.68	54.08
full wiki	Baseline (Yang et al., 2018)	24.68	34.36	5.28	40.98	2.54	17.73
	TF-IDF + Reader	27.55	36.58	10.75	42.45	7.00	21.47
	MUPPET (sentence-level)	30.20	39.43	16.57	46.13	11.38	26.55
	MUPPET (paragraph-level)	**31.07**	**40.42**	**17.00**	**47.71**	**11.76**	**27.62**

Table 1: Primary results for HotpotQA (dev set). At the top of the table, we compare our Paragraph Reader to the baseline model of Yang et al. (2018) (as of writing this paper, no other published results are available other than the baseline results). At the bottom, we compare the end-to-end performance on the full wiki setting. TF-IDF + Reader refers to using the TF-IDF based retriever without our MIPS retriever. MUPPET (sentence-level) refers to our approach with sentence-level representations, and MUPPET (paragraph-level) refers to our approach with paragraph-level representations. For both sentence- and paragraph-level results, we set $n_1 = 32$ and $n_2 = 512$.

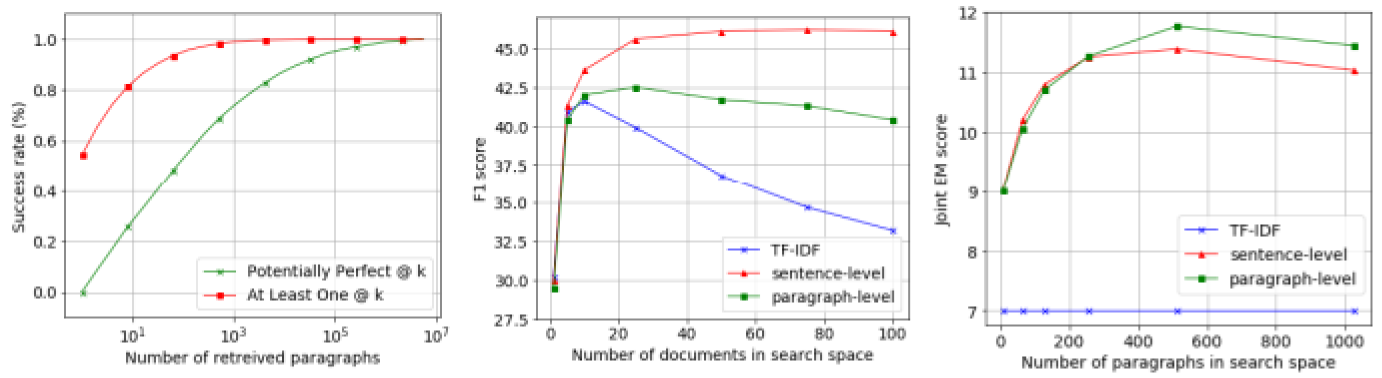


Method	EM	F_1
DrQA (Chen et al., 2017)	28.4	-
DrQA (Chen et al., 2017) (multitask)	29.8	-
R^3 (Wang et al., 2018a)	29.1	37.5
DS-QA (Lin et al., 2018)	28.7	36.6
Par. Ranker + Full Agg. (Lee et al., 2018)	30.2	-
Minimal (Min et al., 2018)	34.7	42.6
Multi-step (Das et al., 2019)	31.9	39.2
BERTserini (Yang et al., 2019)	38.6	46.1
TF-IDF + Reader	34.6	41.6
MUPPET (sentence-level)	**39.3**	**46.2**
MUPPET (paragraph-level)	35.6	42.5

Table 2: Primary results for SQuAD-Open.

  - In SQuAD, each paragraph serves as the context of several questions.
  - This leads to questions being asked about facts less essential to the gist of the paragraph, and thus they would not be encapsulated in a single paragraph representation.
  - In HotpotQA, most of the paragraphs in the training set serve as the context of at most one question.

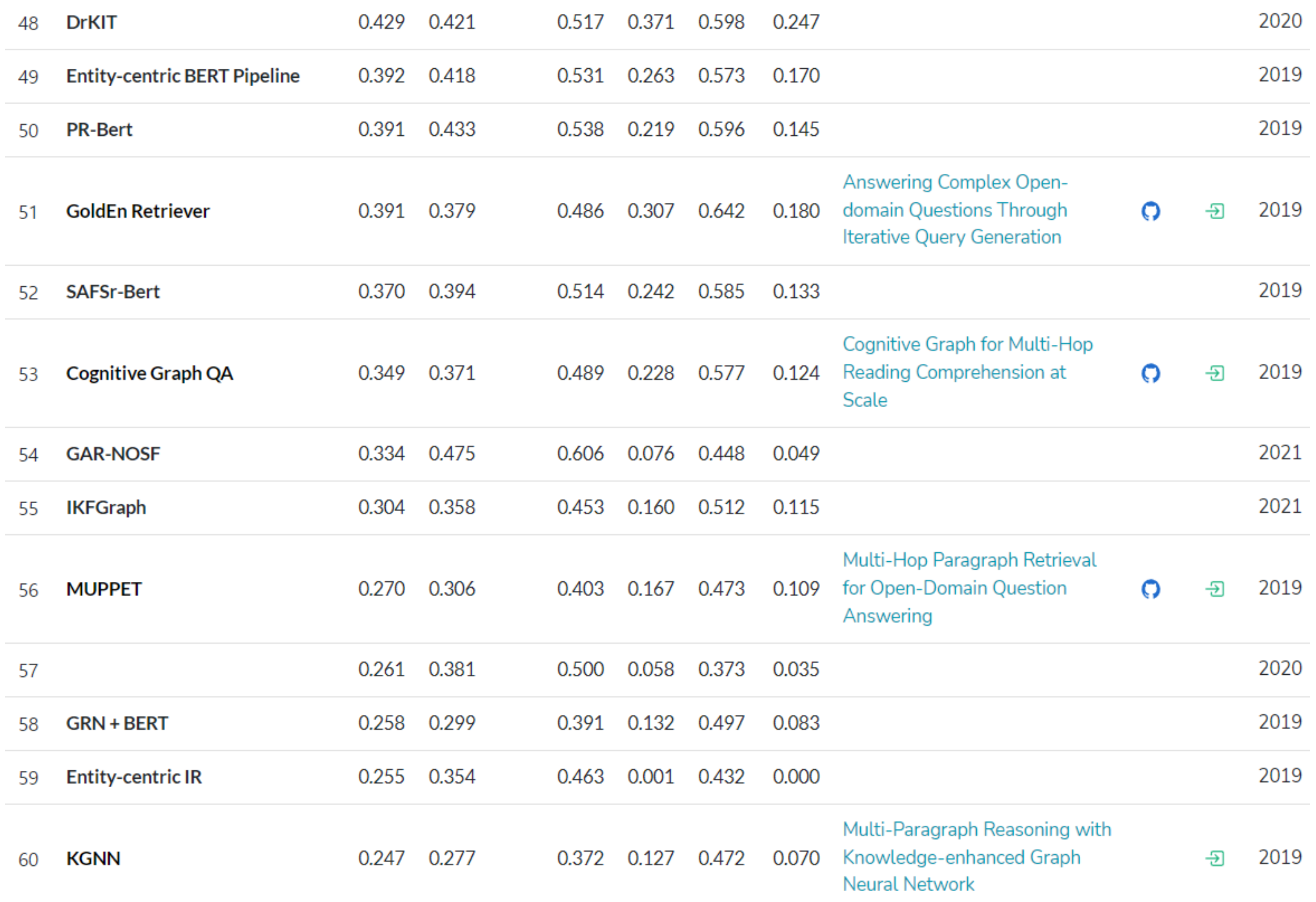
Multi-Hop Paragraph Retrieval for Open-Domain Question Answering [ACL 2019]



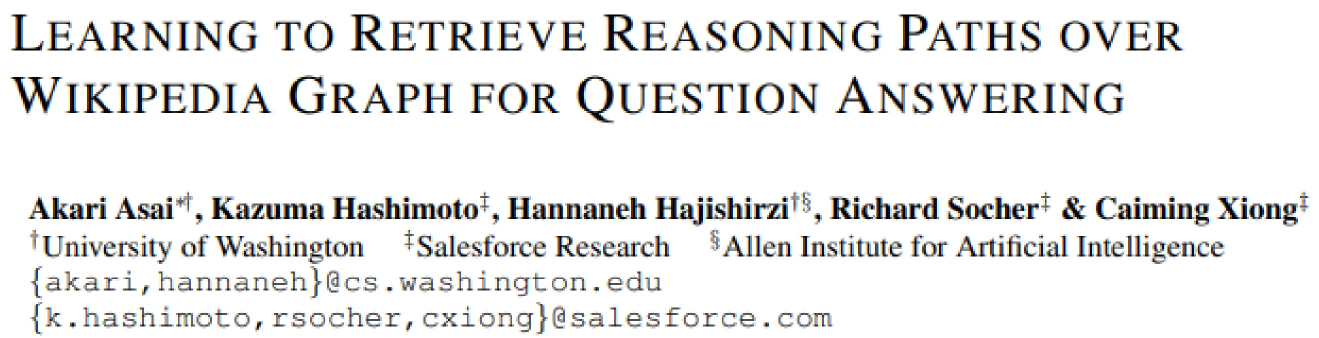
(a) TF-IDF retrieval results (b) SQuAD-Open (c) HotpotQA

Figure 5: Various results based on the TF-IDF retriever. (a) Retrieval results of the TF-IDF hueristic retriever on HotpotQA. *At Least One @ k* is the number of questions for which at least one of the paragraphs containing the supporting facts is retrieved in the top-k paragraphs. *Potentially Perfect @ k* is the number of questions for which both of the paragraphs containing the supporting facts are retrieved in the top-k paragraphs. (b) and (c) Performance analysis on the SQuAD-Open and HotpotQA datasets, respectively, as more documents/paragraphs are retrieved by the TF-IDF heuristic retriever. Note that for SQuAD-Open each document contains several paragraphs, and the reader is fed the top-k TF-IDF ranked paragraphs from within the documents in the search space.

Multi-Hop Paragraph Retrieval for Open-Domain Question Answering [ACL 2019]



48	**DrKIT**	0.429	0.421	0.517	0.371	0.598	0.247		2020
49	**Entity-centric BERT Pipeline**	0.392	0.418	0.531	0.263	0.573	0.170		2019
50	**PR-Bert**	0.391	0.433	0.538	0.219	0.596	0.145		2019
51	**GoldEn Retriever**	0.391	0.379	0.486	0.307	0.642	0.180	Answering Complex Open-domain Questions Through Iterative Query Generation	2019
52	**SAFSr-Bert**	0.370	0.394	0.514	0.242	0.585	0.133		2019
53	**Cognitive Graph QA**	0.349	0.371	0.489	0.228	0.577	0.124	Cognitive Graph for Multi-Hop Reading Comprehension at Scale	2019
54	**GAR-NOSF**	0.334	0.475	0.606	0.076	0.448	0.049		2021
55	**IKFGraph**	0.304	0.358	0.453	0.160	0.512	0.115		2021
56	**MUPPET**	0.270	0.306	0.403	0.167	0.473	0.109	Multi-Hop Paragraph Retrieval for Open-Domain Question Answering	2019
57		0.261	0.381	0.500	0.058	0.373	0.035		2020
58	**GRN + BERT**	0.258	0.299	0.391	0.132	0.497	0.083		2019
59	**Entity-centric IR**	0.255	0.354	0.463	0.001	0.432	0.000		2019
60	**KGNN**	0.247	0.277	0.372	0.127	0.472	0.070	Multi-Paragraph Reasoning with Knowledge-enhanced Graph Neural Network	2019



Learning to Retrieve Reasoning Paths over Wikipedia Graph for Question Answering

Akari Asai[*†], **Kazuma Hashimoto**[‡], **Hannaneh Hajishirzi**[†§], **Richard Socher**[‡] **& Caiming Xiong**[‡]
[†]University of Washington [‡]Salesforce Research [§]Allen Institute for Artificial Intelligence
{akari,hannaneh}@cs.washington.edu
{k.hashimoto,rsocher,cxiong}@salesforce.com

ICLR 2020

Learning to Retrieve Reasoning Paths over Wikipedia Graph for Question Answering [ICLR 2020]

- Recent open-domain QA methods learn end-to-end models to jointly retrieve and read documents
    - The methods poposed by *Seo et al., 2019* and *Lee et al., 2019*, face challenges for **entity-centric questions** since compressing the necessary information into an embedding space does not capture **lexical information** in entities.
    - Also, although Cognitive Graph(Ding et al., 2019) incorporates **entity links** between documents for multi-hop QA, they compile a fixed list of documents independently and expects the reader to find the reasoning paths.
    - Multi-step reasoner (Das et al., 2019) and MUPPET (Feldman & El-Yaniv., 2019) use multi-step retrievers, but they do not use graph structure of documents during iterative retrieval process and do not accommodate arbitrary steps of reasoning.

    - The authors introduce a new **recurrent graph-based retrieval method** that learns to retrieve evidence documents as reasoning paths for answering questions.
    - The proposed method leverages an existing reading comprehension model to answer questions by ranking the retrieved reasoning paths.
    - It leverages **Wikipedia graph** to retrieve documents that are **lexically or semantically** distant to questions, and is adaptive to any reasoning path lengths.
    - The **strong interplay between the retriever model and reader model** enables the entire method to answer complex questions.

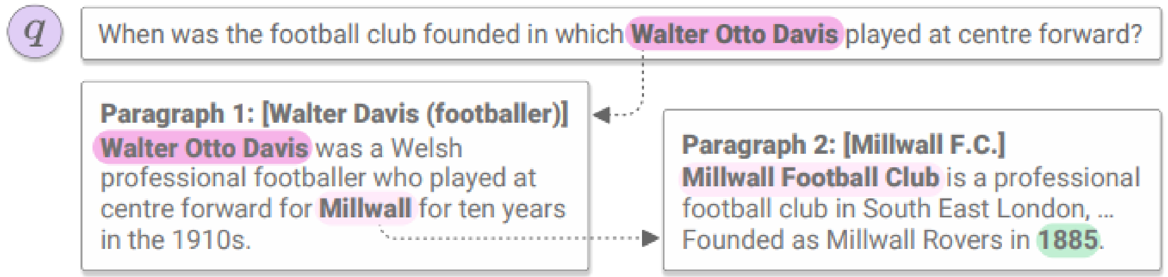

q
When was the football club founded in which Walter Otto Davis played at centre forward?
Paragraph 1: [Walter Davis (footballer)]
Walter Otto Davis was a Welsh professional footballer who played at centre forward for Millwall for ten years in the 1910s.
Paragraph 2: [Millwall F.C.]
Millwall Football Club is a professional football club in South East London, ... Founded as Millwall Rovers in 1885.

Figure 1: An example of open-domain multi-hop question from HotpotQA. Paragraph 2 is unlikely to be retrieved using TF-IDF retrievers due to little lexical overlap to the given question.

Learning to Retrieve Reasoning Paths over Wikipedia Graph for Question Answering [ICLR 2020]

- Model Architecture

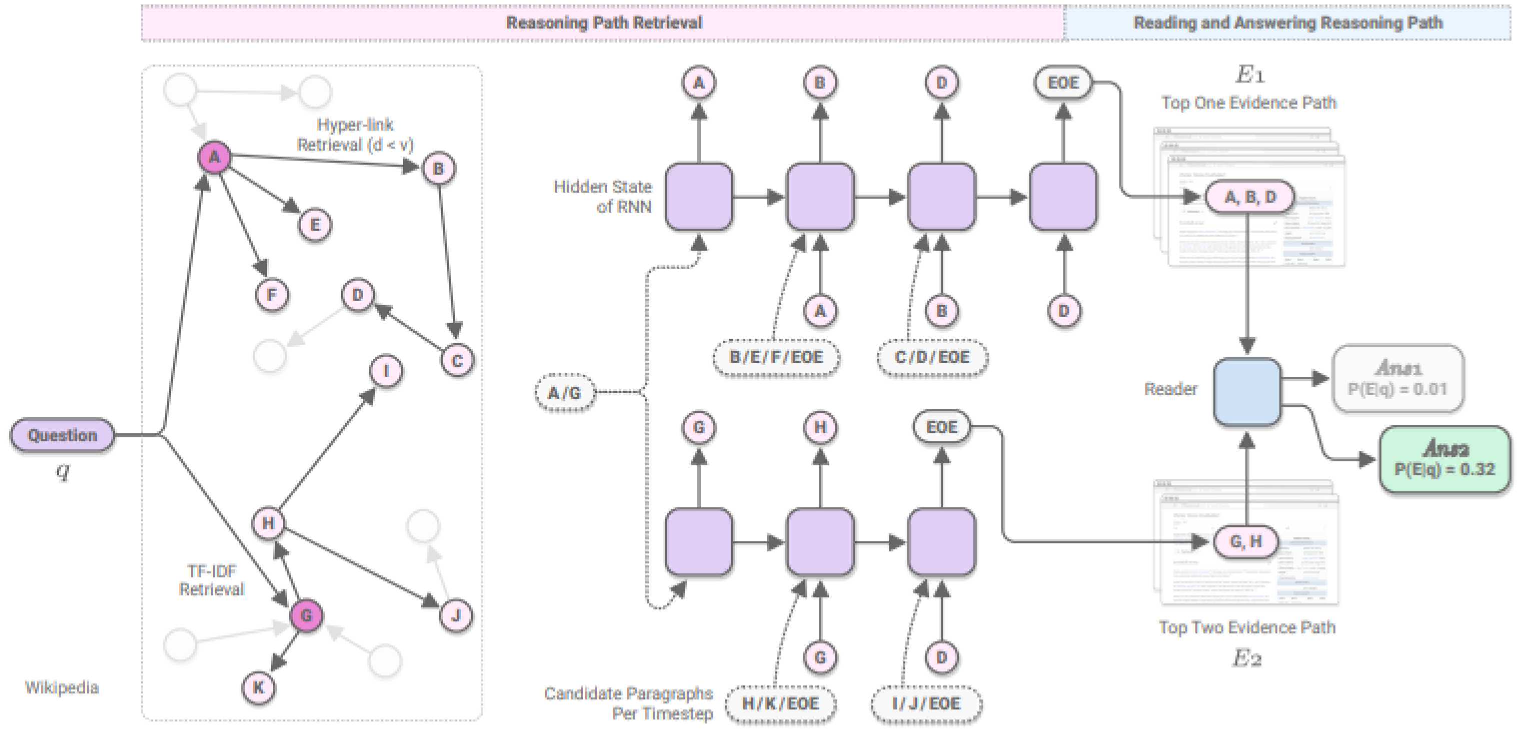


Figure 2: Overview of our framework.

Learning to Retrieve Reasoning Paths over Wikipedia Graph for Question Answering [ICLR 2020]

- Model Architecture
  - Evidence paragraphs for a complex question do not necessarily have **lexical overlaps** with the question, but one of them is likely to be retrieved, and its entity mentions and question often entail another paragraph.
  - The authors first construct a **graph of paragraphs** covering all Wikipedia paragraphs.

  - **Hyperlinks** are commonly used to construct relationships between articles on the web, usually maintained by article writers, and are thus useful knowledge resources.
  - The authors use hyperlinks to construct direct edges within the Wikipedia graph.

  - The authors use a RNN to **model the reasoning paths** for the question.
  - At the t-th time step, the model selects a paragraph p_i among candidate paragraphs C_t given the current hidden state h_t of the RNN.
  - The initial hidden state use BERT's [CLS] token representation, and then compute probability $P(p_i|h_t)$ where p_i is selected.
  - This RNN selection procedure captures relationships between paragraphs in the reasoning path by conditioning on the selection history.

$$w_i = \mathrm{BERT}_{[\mathrm{CLS}]}(q, p_i) \in \mathbb{R}^d, \tag{2}$$

$$P(p_i|h_t) = \sigma(w_i \cdot h_t + b), \tag{3}$$

$$h_{t+1} = \mathrm{RNN}(h_t, w_i) \in \mathbb{R}^d, \tag{4}$$

  - The next candidate set C_{t+1} is constructed using the paragraphs that are linked from the selected paragraph p_i in the Wikipedia graph.
  - To allow the model to flexibly retrieve multiple paragraphs within C_t, the authors also add K-best paragraphs other than p_i to C_{t+1}.
  - Also, **beam search** is used to explore paths in the directed graph where the final probability score would results in $P(p_i|h_1) \dots P(p_k|h_{|E|})$

Learning to Retrieve Reasoning Paths over Wikipedia Graph for Question Answering [ICLR 2020]

- Training Procedure
  - The retriever is trained in a supervised fashion using evidence paragraphs annotated for each question.
  - The authors augment a ground-truth reasoning path $g = [p_1, \ldots, p_g]$ using available annotated data in each dataset.
  - The authors also use negative examples along with the ground-truth paragraphs, (1) **TF-IDF-based** and (2) **hyperlink-based**.
  - The number of negative examples is typically set to 50.

  - For the sequential prediction task, the authors estimate $P(p_i|h_t)$ independently using **binary cross-entropy loss** to maximize probability values of all possible paths.

$$L_{\text{retr}}(p_t, h_t) = -\log P(p_t|h_t) - \sum_{\tilde{p} \in \tilde{\mathbf{C}}_t} \log(1 - P(\tilde{p}|h_t)), \qquad (5)$$

  - The reader model first verifies each reasoning path in $E = \{E_1, \ldots, E_B\}$, and finally outputs an answer span a from the most plausible reasoning path.
  - The reader is modeled as a **multi-task learning** of (1) **reading comprehension** and (2) **reasoning path re-ranking**.
  - For reading comprehension, BERT is used where the input is the concatenation of the question text and the text of all the paragraphs in E.

$$P(E|q) = \sigma(w_n \cdot u_E) \quad \text{s.t.} \quad u_E = \text{BERT}_{[\text{CLS}]}(q, E) \in \mathbb{R}^D, \qquad (6)$$

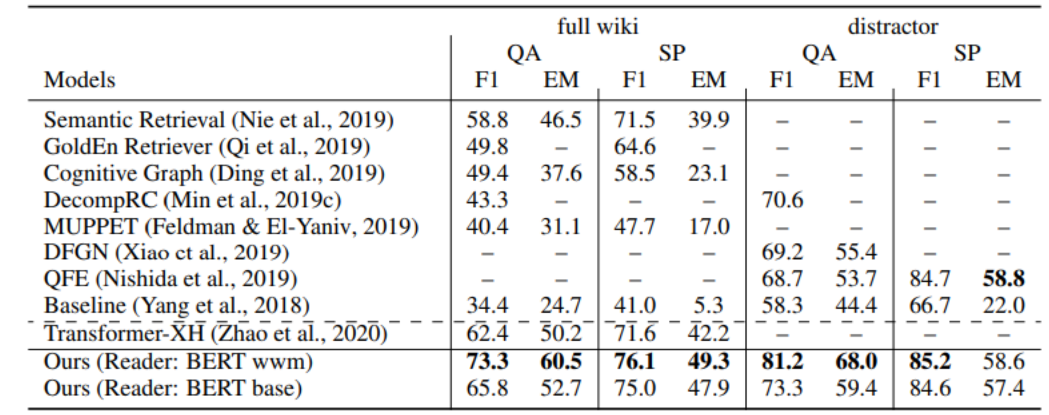
$$E_{best} = \underset{E \in \mathbf{E}}{\arg\max}\, P(E|q), \quad S_{\text{read}} = \underset{i,j,\; i \leq j}{\arg\max}\, P_i^{start} P_j^{end}, \qquad (7)$$

  - The objective is the sum of cross entropy losses for the span prediction and re-ranking tasks.

$$L_{\text{read}} = L_{\text{span}} + L_{\text{no\_answer}} = (-\log P_{y^{start}}^{start} - \log P_{y^{end}}^{end}) - \log P^r, \qquad (8)$$

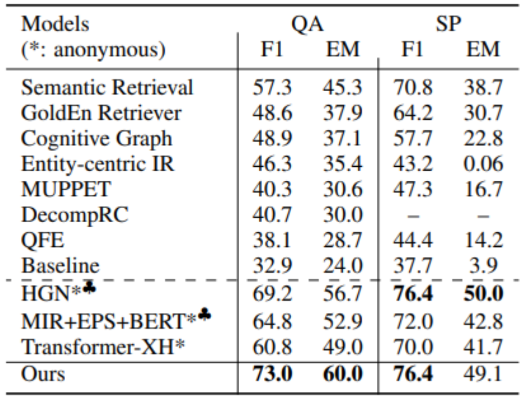
Learning to Retrieve Reasoning Paths over Wikipedia Graph for Question Answering [ICLR 2020]

- Experiments / Results
  - The authors use HotpotQA, SQuAD Open, Natural Questions Open to test the performance, measured by EM and F1 Score.



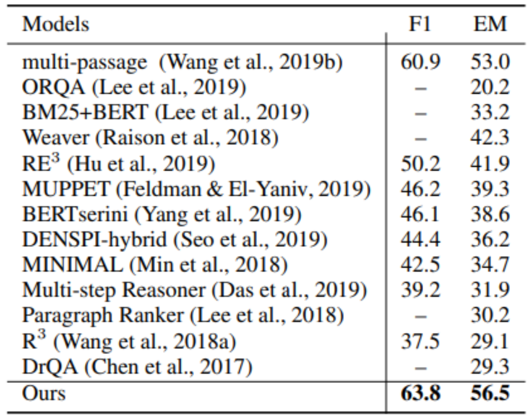
	full wiki				distractor			
	QA		SP		QA		SP	
Models	F1	EM	F1	EM	F1	EM	F1	EM
Semantic Retrieval (Nie et al., 2019)	58.8	46.5	71.5	39.9	–	–	–	–
GoldEn Retriever (Qi et al., 2019)	49.8	–	64.6	–	–	–	–	–
Cognitive Graph (Ding et al., 2019)	49.4	37.6	58.5	23.1	–	–	–	–
DecompRC (Min et al., 2019c)	43.3	–	–	–	70.6	–	–	–
MUPPET (Feldman & El-Yaniv, 2019)	40.4	31.1	47.7	17.0	–	–	–	–
DFGN (Xiao et al., 2019)	–	–	–	–	69.2	55.4	–	–
QFE (Nishida et al., 2019)	–	–	–	–	68.7	53.7	84.7	**58.8**
Baseline (Yang et al., 2018)	34.4	24.7	41.0	5.3	58.3	44.4	66.7	22.0
Transformer-XH (Zhao et al., 2020)	62.4	50.2	71.6	42.2	–	–	–	–
Ours (Reader: BERT wwm)	**73.3**	**60.5**	**76.1**	**49.3**	**81.2**	**68.0**	**85.2**	58.6
Ours (Reader: BERT base)	65.8	52.7	75.0	47.9	73.3	59.4	84.6	57.4

Table 1: **HotpotQA development set results**: QA and SP (supporting fact prediction) results on HotpotQA's full wiki and distractor settings. "–" denotes no results are available.



Models	QA		SP	
(*: anonymous)	F1	EM	F1	EM
Semantic Retrieval	57.3	45.3	70.8	38.7
GoldEn Retriever	48.6	37.9	64.2	30.7
Cognitive Graph	48.9	37.1	57.7	22.8
Entity-centric IR	46.3	35.4	43.2	0.06
MUPPET	40.3	30.6	47.3	16.7
DecompRC	40.7	30.0	–	–
QFE	38.1	28.7	44.4	14.2
Baseline	32.9	24.0	37.7	3.9
HGN*♣	69.2	56.7	**76.4**	**50.0**
MIR+EPS+BERT*♣	64.8	52.9	72.0	42.8
Transformer-XH*	60.8	49.0	70.0	41.7
Ours	**73.0**	**60.0**	**76.4**	49.1

Table 2: **HotpotQA full wiki test set results**: official leaderboard results (on November 6, 2019) on the hidden test set of the HotpotQA full wiki setting. Work marked with ♣ appeared after September 25.

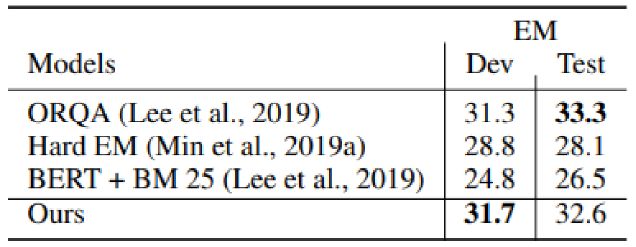


Models	F1	EM
multi-passage (Wang et al., 2019b)	60.9	53.0
ORQA (Lee et al., 2019)	–	20.2
BM25+BERT (Lee et al., 2019)	–	33.2
Weaver (Raison et al., 2018)	–	42.3
RE^3 (Hu et al., 2019)	50.2	41.9
MUPPET (Feldman & El-Yaniv, 2019)	46.2	39.3
BERTserini (Yang et al., 2019)	46.1	38.6
DENSPI-hybrid (Seo et al., 2019)	44.4	36.2
MINIMAL (Min et al., 2018)	42.5	34.7
Multi-step Reasoner (Das et al., 2019)	39.2	31.9
Paragraph Ranker (Lee et al., 2018)	–	30.2
R^3 (Wang et al., 2018a)	37.5	29.1
DrQA (Chen et al., 2017)	–	29.3
Ours	**63.8**	**56.5**

Table 3: **SQuAD Open results**: we report F1 and EM scores on the test set of SQuAD Open, following previous work.

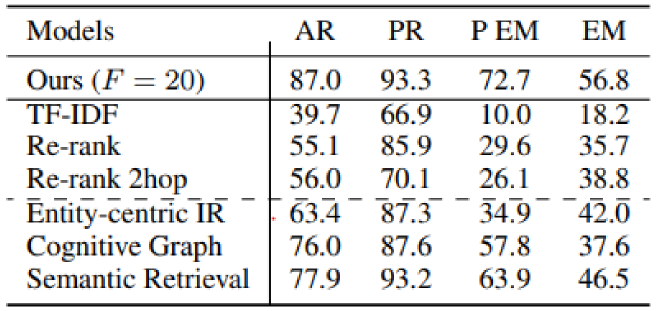
Learning to Retrieve Reasoning Paths over Wikipedia Graph for Question Answering [ICLR 2020]

- Experiments / Results
  - The authors use HotpotQA, SQuAD Open, Natural Questions Open to test the performance, measured by EM and F1 Score.



Models	EM Dev	EM Test
ORQA (Lee et al., 2019)	31.3	**33.3**
Hard EM (Min et al., 2019a)	28.8	28.1
BERT + BM 25 (Lee et al., 2019)	24.8	26.5
Ours	**31.7**	32.6

Table 4: **Natural Questions Open results**: we report EM scores on the test and development sets of Natural Questions Open, following previous work.

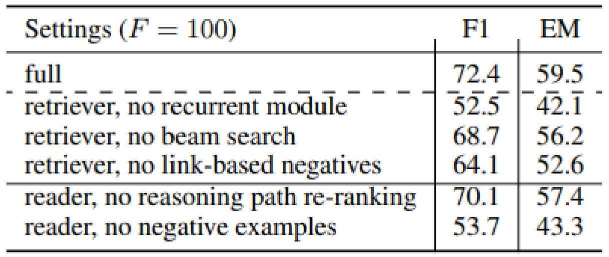


Models	AR	PR	P EM	EM
Ours ($F = 20$)	87.0	93.3	72.7	56.8
TF-IDF	39.7	66.9	10.0	18.2
Re-rank	55.1	85.9	29.6	35.7
Re-rank 2hop	56.0	70.1	26.1	38.8
Entity-centric IR	63.4	87.3	34.9	42.0
Cognitive Graph	76.0	87.6	57.8	37.6
Semantic Retrieval	77.9	93.2	63.9	46.5

Table 5: **Retrieval evaluation**: Comparing our retrieval method with other methods across Answer Recall, Paragraph Recall, Paragraph EM, and QA EM metrics.

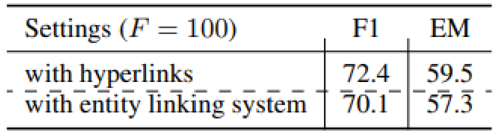
Learning to Retrieve Reasoning Paths over Wikipedia Graph for Question Answering [ICLR 2020]

- Ablation Studies
  - The most critical component in **the retriever model** is the recurrent module, dropping the EM by 17.4 points.
  - Performance drop by removing the reasoning path re-ranking indicates the importance of verifying reasoning paths **in the reader**.



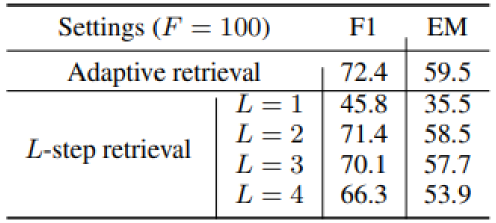
Settings ($F = 100$)	F1	EM
full	72.4	59.5
retriever, no recurrent module	52.5	42.1
retriever, no beam search	68.7	56.2
retriever, no link-based negatives	64.1	52.6
reader, no reasoning path re-ranking	70.1	57.4
reader, no negative examples	53.7	43.3

Table 6: **Ablation study**: evaluating different variants of our model on HotpotQA full wiki.



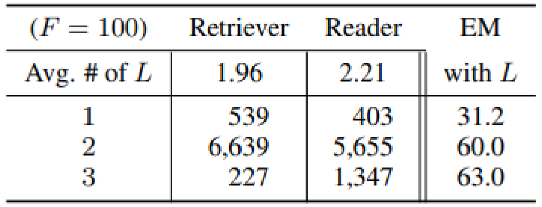
Settings ($F = 100$)	F1	EM
with hyperlinks	72.4	59.5
with entity linking system	70.1	57.3

Table 7: **Performance with different link structures**: comparing our results on the Hotpot QA full wiki development set when we use an off-the-shelf entity linking system instead of the Wikipedia hyperlinks.



Settings ($F = 100$)		F1	EM
Adaptive retrieval		72.4	59.5
L-step retrieval	$L = 1$	45.8	35.5
	$L = 2$	71.4	58.5
	$L = 3$	70.1	57.7
	$L = 4$	66.3	53.9

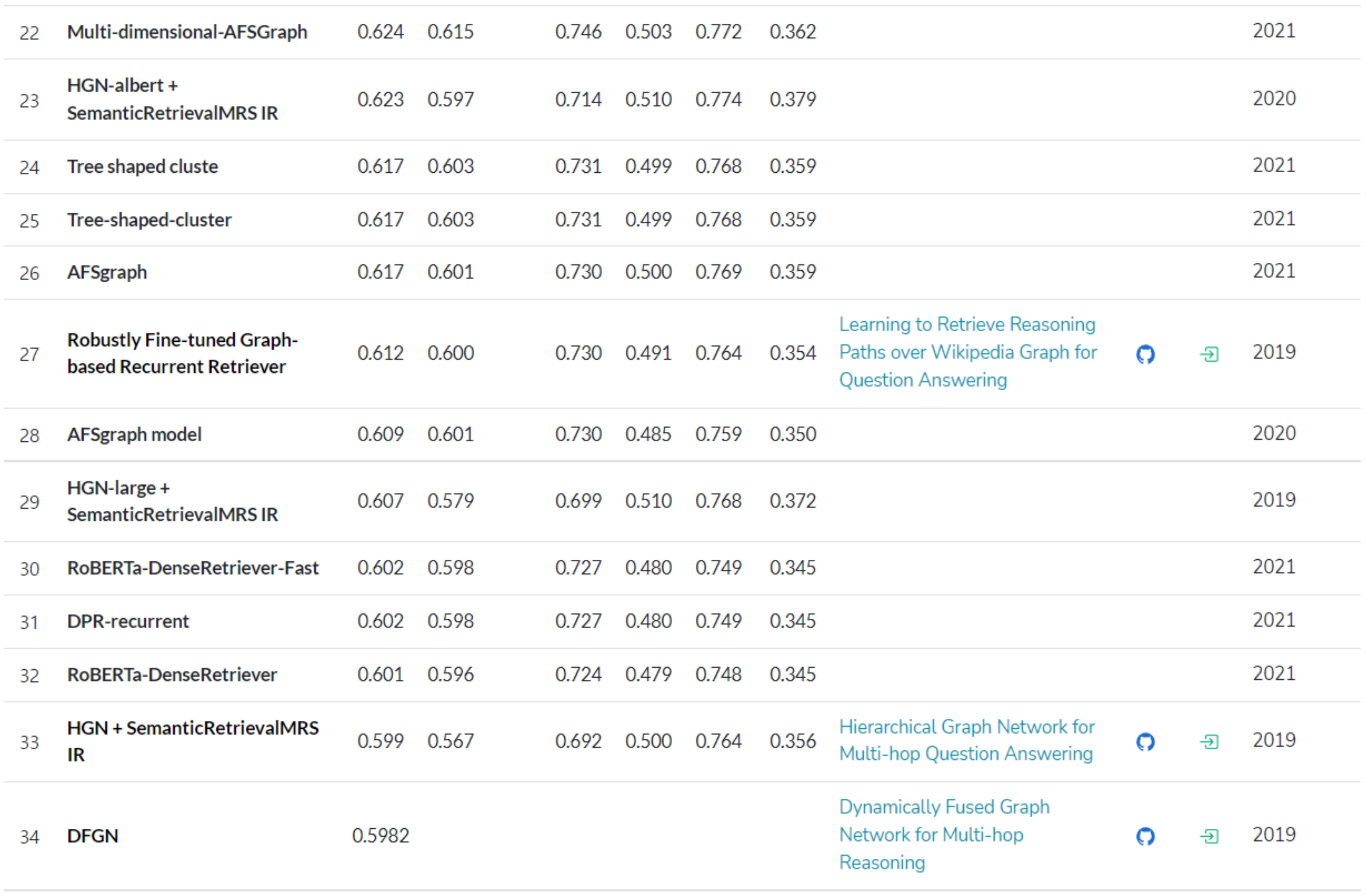
Table 8: **Performance with different reasoning path length**: comparing the performance with different path length on HotpotQA full wiki. L-step retrieval sets the number of the reasoning steps to a fixed number.



($F = 100$)	Retriever	Reader	EM
Avg. # of L	1.96	2.21	with L
1	539	403	31.2
2	6,639	5,655	60.0
3	227	1,347	63.0

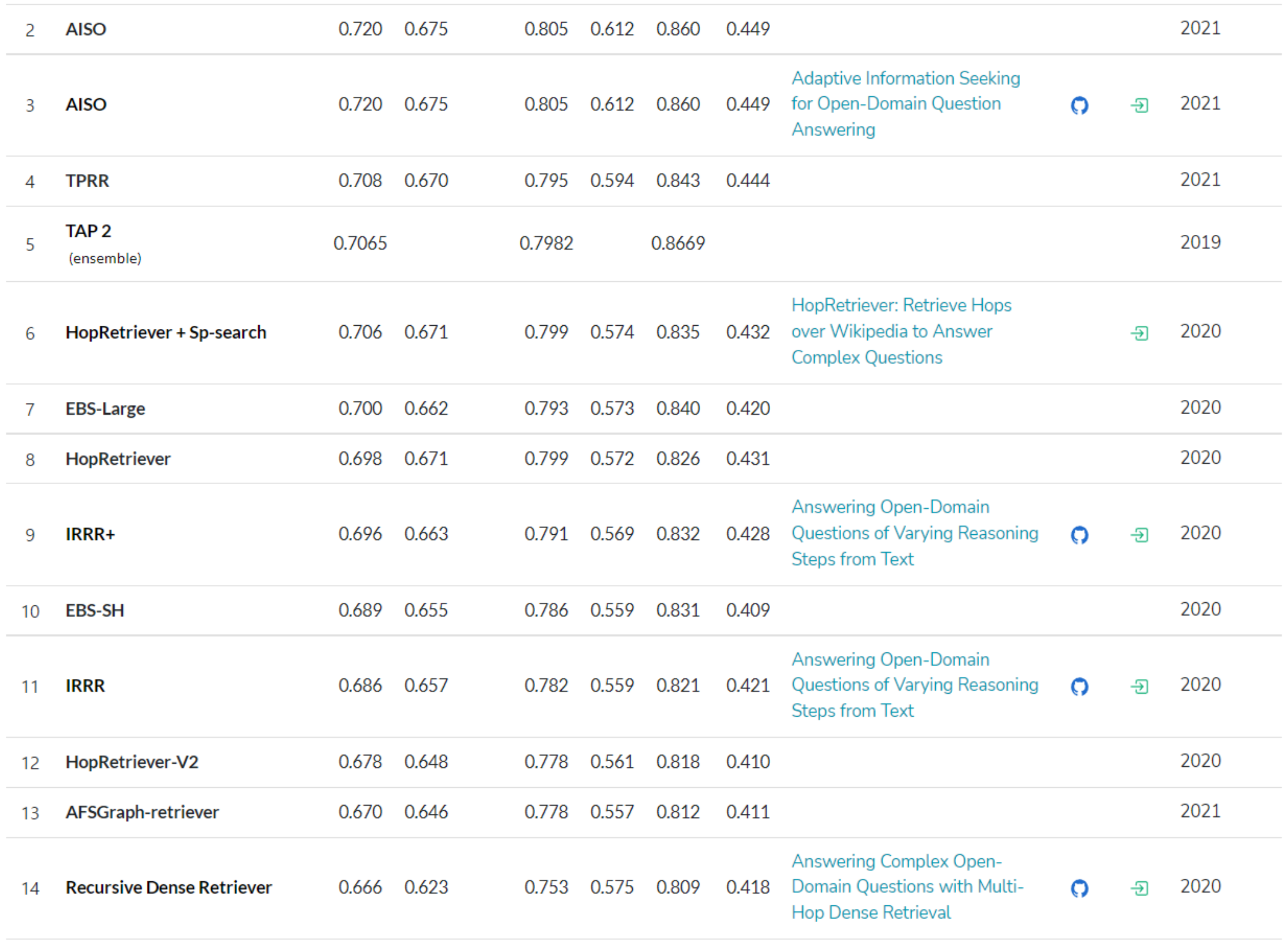
Table 9: **Statistics of the reasoning paths**: the average length and the distribution of length of the reasoning paths selected by our retriever and reader for HotpotQA full wiki. Avg. EM represents QA EM performance.

Learning to Retrieve Reasoning Paths over Wikipedia Graph for Question Answering [ICLR 2020]



22	Multi-dimensional-AFSGraph	0.624	0.615		0.746	0.503	0.772	0.362				2021
23	HGN-albert + SemanticRetrievalMRS IR	0.623	0.597		0.714	0.510	0.774	0.379				2020
24	Tree shaped cluste	0.617	0.603		0.731	0.499	0.768	0.359				2021
25	Tree-shaped-cluster	0.617	0.603		0.731	0.499	0.768	0.359				2021
26	AFSgraph	0.617	0.601		0.730	0.500	0.769	0.359				2021
27	Robustly Fine-tuned Graph-based Recurrent Retriever	0.612	0.600		0.730	0.491	0.764	0.354	Learning to Retrieve Reasoning Paths over Wikipedia Graph for Question Answering	●	→	2019
28	AFSgraph model	0.609	0.601		0.730	0.485	0.759	0.350				2020
29	HGN-large + SemanticRetrievalMRS IR	0.607	0.579		0.699	0.510	0.768	0.372				2019
30	RoBERTa-DenseRetriever-Fast	0.602	0.598		0.727	0.480	0.749	0.345				2021
31	DPR-recurrent	0.602	0.598		0.727	0.480	0.749	0.345				2021
32	RoBERTa-DenseRetriever	0.601	0.596		0.724	0.479	0.748	0.345				2021
33	HGN + SemanticRetrievalMRS IR	0.599	0.567		0.692	0.500	0.764	0.356	Hierarchical Graph Network for Multi-hop Question Answering	●	→	2019
34	DFGN	0.5982							Dynamically Fused Graph Network for Multi-hop Reasoning	●	→	2019

Learning to Retrieve Reasoning Paths over Wikipedia Graph for Question Answering [ICLR 2020]



2	**AISO**	0.720	0.675	0.805	0.612	0.860	0.449				2021
3	**AISO**	0.720	0.675	0.805	0.612	0.860	0.449	Adaptive Information Seeking for Open-Domain Question Answering	●	↪	2021
4	**TPRR**	0.708	0.670	0.795	0.594	0.843	0.444				2021
5	**TAP 2** (ensemble)	0.7065		0.7982		0.8669					2019
6	**HopRetriever + Sp-search**	0.706	0.671	0.799	0.574	0.835	0.432	HopRetriever: Retrieve Hops over Wikipedia to Answer Complex Questions		↪	2020
7	**EBS-Large**	0.700	0.662	0.793	0.573	0.840	0.420				2020
8	**HopRetriever**	0.698	0.671	0.799	0.572	0.826	0.431				2020
9	**IRRR+**	0.696	0.663	0.791	0.569	0.832	0.428	Answering Open-Domain Questions of Varying Reasoning Steps from Text	●	↪	2020
10	**EBS-SH**	0.689	0.655	0.786	0.559	0.831	0.409				2020
11	**IRRR**	0.686	0.657	0.782	0.559	0.821	0.421	Answering Open-Domain Questions of Varying Reasoning Steps from Text	●	↪	2020
12	**HopRetriever-V2**	0.678	0.648	0.778	0.561	0.818	0.410				2020
13	**AFSGraph-retriever**	0.670	0.646	0.778	0.557	0.812	0.411				2021
14	**Recursive Dense Retriever**	0.666	0.623	0.753	0.575	0.809	0.418	Answering Complex Open-Domain Questions with Multi-Hop Dense Retrieval	●	↪	2020

ANY QUESTIONS?