Open Domain Question Answering (Part 3)

(Papers published in 2021)

Department of Computer Science, Yonsei University

Seungone Kim

louisdebroglie@yonsei.ac.kr

Referenced Papers

- Prerequisites / Additional Papers
  - Matching the Blanks : Distributional Similarity for Relation Learning [ACL 2019]
  - Cognitive Graph for Multi-Hop Reading Comprehension at Scale [ACL 2019]
  - Differentiable Reasoning over a Virtual Knowledge Base [ICLR 2020]
  - Revealing the Importance of Semantic Retrieval for Machine Reading at Scale [EMNLP-IJCNLP 2019]
  - Multi-step Entity-centric Information Retrieval for Multi-Hop Question Answering [EMNLP 2019 Workshop]

  - Language Models as Knowledge Bases? [EMNLP-IJCNLP 2019]
  - How much knowledge can you pack into the parameters of a language model? [EMNLP 2020]
  - oLMpics-On what Language Model Pre-training Captures [TACL 2020]
  - How can we know what language models know? [TACL 2020]

- Key Papers
  - HopRetriever : Retrieve hops over Wikipedia to Answer Complex Questions [AAAI 2021]
  - Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering [EACL 2021]

HopRetriever: Retrieve Hops over Wikipedia to Answer Complex Questions

Shaobo Li,[1*] Xiaoguang Li,[2] Lifeng Shang,[2] Xin Jiang,[2]
Qun Liu,[2] Chengjie Sun,[1] Zhenzhou Ji,[1] Bingquan Liu[1]
[1]Harbin Institute of Technology
[2]Huawei Noah's Ark Lab
shli@insun.hit.edu.cn, {lixiaoguang11, shang.lifeng, Jiang.Xin, qun.liu}@huawei.com,
{sunchengjie, jizhenzhou, liubq}@hit.edu.cn

AAAI 2021

HopRetriever : Retrieve Hops over Wikipedia to Answer Complex Questions [AAAI 2021]

- Collecting supporting evidence from large corpora of text is of great challenge for Open Domain QA, especially for Multi-hop Open Domain QA.
  - Most of the recent works view multi-hop evidence collection as an **iterative document retrieval problem**. (Asai et al., 2020; Das et al., 2019a)
  - In contrast, some others focus on mentioned entities and try to traverse textual data like a **structured KB**. (Dhingra et al., 2020; Ding et al., 2019)
  - The authors argue that 1) **unstructured facts** inside a document and 2) **structured and implicit relations** between entities are both needed.
  - To collect sufficient evidence, it's necessary to consider both **relational structures** between entities and **unstructured knowledge** hidden inside a document.

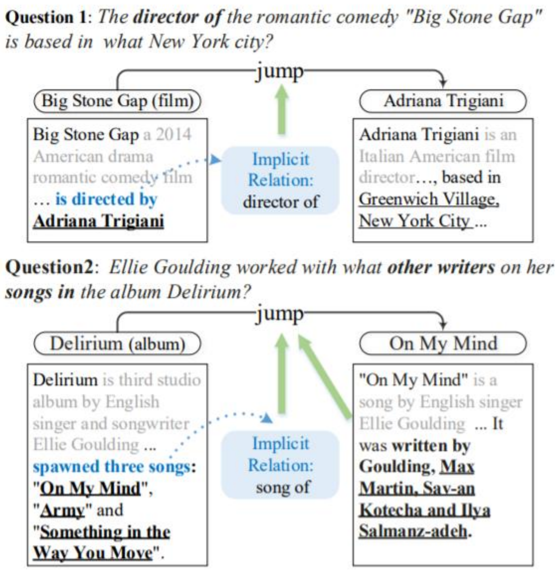


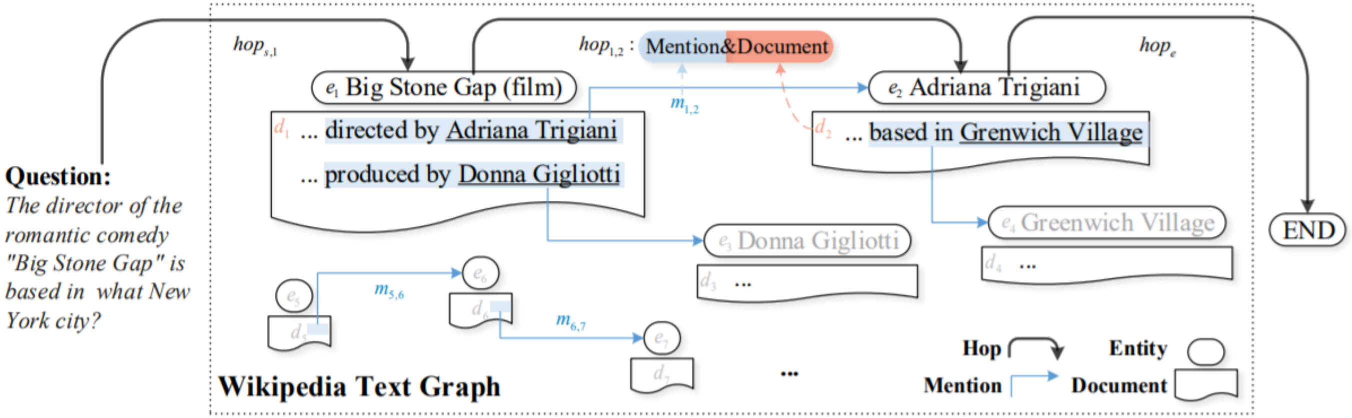
Figure 1: Two examples showing that both structured relation and unstructured fact are needed for complex question answering.

HopRetriever : Retrieve Hops over Wikipedia to Answer Complex Questions [AAAI 2021]

- Intuition
  - The authors define a **hop** as the combination of a hyperlink and a corresponding outbound link document.
  - A **hyperlink** in Wikipedia implies how the introductory document of an entity mentions some other, while the **outbound link document** stores all the unstructured facts and events, which makes a hop contain both relational and factoid evidence for future retrieval.

  - One challenge is how to transform the **binary hyperlink** in Wikipedia to **distributed representations** implying the implicit and complicated entity relation.
  - One step towards this is the recent work on **distributed relation learning**. (Soares et al., 2019)
  - The relation representations are learned solely from the entity-linked text in an unsupervised way.

  - For each entity mention within Wikipedia documents, the authors encode the textual context around it into **mention embedding** to represent the implicit structured knowledge.



HopRetriever : Retrieve Hops over Wikipedia to Answer Complex Questions [AAAI 2021]

- Methodology
  - The authors propose **HopRetriever** to take place of the retriever model while keeping the answer extraction model as standard.
  - Each Wikipedia page corresponds to an **entity** $\boldsymbol{e_i}$, accompanied by an **introductory document** $\boldsymbol{d_i}$.
  - If there exists an anchor text in d_i linked to e_j, the authors denote it as a **mention** $\boldsymbol{m_{i,j} = e_i \xrightarrow{d_i} e_j}$.
  - Accordingly, the **knowledge source** is formulated as $\boldsymbol{K = \{D, E, M\}}$ *such that* $E = \{e_i\},\ D = \{d_i\},\ M = \{m_{i,j}\}$.

- Hop Encoding
  - The **representation of each hop** consists of **mention embeddings** $\boldsymbol{m_{i,j}}$ that implies the entity relation from e_i to e_j, and the **document embedding** $\boldsymbol{u_j}$ of entity e_j.
  - If e_j is not mentioned directly in the introductory document of e_i, the relation is represented between with a trainable uniformed vector m_p.

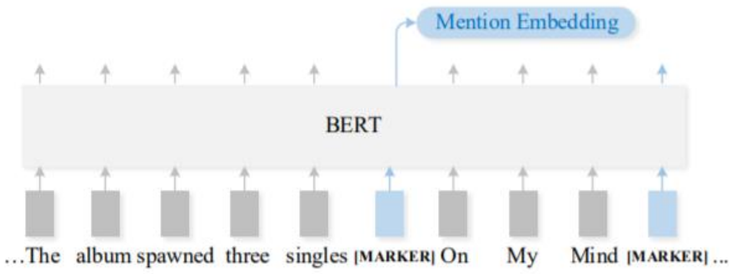


Figure 3: Encoding the mention using entity markers.

$$\mathbf{m}_{i,j} = \begin{cases} \mathrm{BERT}_{[\text{M-}j]}(q; d_i), & \text{if } m_{i,j} \in M \\ \mathbf{m}_{\mathrm{P}}, & \text{otherwise} \end{cases} \quad (3)$$

HopRetriever : Retrieve Hops over Wikipedia to Answer Complex Questions [AAAI 2021]

- Hop Encoding
  - The **representation of each hop** consists of **mention embeddings $m_{i,j}$** that implies the entity relation from e_i to e_j, and the **document embedding u_j** of entity e_j.
  - The unstructured knowledge about entity e_j is encoded as document embedding u_j by feeding the textual facts in d_j with the question q.

$$\mathbf{u}_j = \mathrm{BERT}_{[\mathrm{CLS}]}(q; d_j). \tag{4}$$

  - The mention embedding $m_{i,j}$ and document embedding u_j are fused together as hop encoding $hop_{i,j}$ by attention mechanism.

$$\begin{aligned} a_m &= \mathbf{h}\mathbf{W}_k\mathbf{m}_{i,j} \\ a_u &= \mathbf{h}\mathbf{W}_k\mathbf{u}_j \\ \{w_m, w_u\} &= \mathrm{softmax}(\{a_m, a_u\}) \\ \mathbf{hop}_{i,j} &= w_m \cdot \mathbf{W}_v\mathbf{m}_{i,j} + w_u \cdot \mathbf{W}_v\mathbf{u}_j, \end{aligned} \tag{5}$$

  - In the equation above, h is the retrieval history that acts as a query vector that interacts with key vectors to calculate the importance weight for mention embedding.

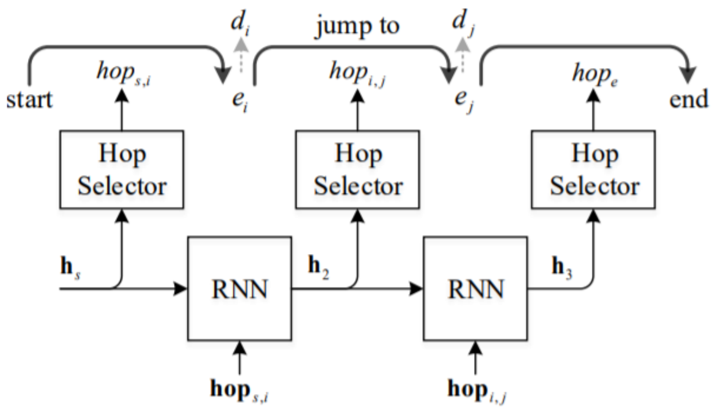
HopRetriever : Retrieve Hops over Wikipedia to Answer Complex Questions [AAAI 2021]

- Iterative Retrieval of Hops
  - **Probability of retrieving the document** at t step is calculated by the dot product of the retrieval history vector h and the hop encoding vector $hop_{i,j}$.
  - The retrieval history vector h is acquired using a RNN.

$$\mathbf{h}_t = \begin{cases} \mathbf{h}_s, & t = 1 \\ \mathrm{RNN}(\mathbf{h}_{t-1}, \mathbf{hop}_{k,i}), & t \geq 2 \end{cases} \qquad (7)$$

Notation	Encoding	Explanation
$hop_{i,j}$	$f(\mathbf{m}_P, \mathbf{u}_j)$	e_j is not mentioned in d_i
	$f(\mathbf{m}_{i,j}, \mathbf{u}_j)$	e_j is mentioned in d_i
$hop_{s,j}$	$f(\mathbf{m}_P, \mathbf{u}_j)$	Select d_j at the beginning
hop_e	$f(\mathbf{m}_P, \mathbf{u}_e)$	Retrieval finish

Table 1: Types of hop encoding.



- Fine-Grained Sentence-Level Retrieval
  - In HopRetriever, the **supporting sentence prediction** is added as an auxiliary task along with the primary hop retrieval task.
  - At step t, the probability $p(s_{i,l})$ that indicates the l-th sentence in the latest retrieved document d_i is a supporting sentence.

$$\mathbf{s}_{i,l} = \mathrm{BERT}_{[\text{SM-}l]}(q; d_i) \qquad (8)$$
$$p(s_{i,l}) = \mathrm{sigmoid}(\mathbf{h}_t \mathbf{W}_s \mathbf{s}_{i,l}), \qquad (9)$$

  - If $p(s_{i,l}) > 0.5$, then the l-th sentence in the document d_i is identified as a supporting sentence.

HopRetriever : Retrieve Hops over Wikipedia to Answer Complex Questions [AAAI 2021]

- Objective Functions of HopRetriever
  - HopRetriever is a **sequence prediction** model with binary cross-entropy objective at each step, along with the auxiliary **supporting sentence prediction** task.

$$\log p(d_j) + \sum_{\bar{d}_j \in D, \bar{d}_j \neq d_j} \log(1 - p(\bar{d}_j)), \qquad (10)$$

$$\sum_{l \in L_i} \log p(s_{i,l}) + \sum_{l \notin L_i} \log(1 - p(s_{i,l})), \qquad (11)$$

- Experiments
  - HopRetriever is evaluated on HotpotQA.
  - The pipeline consists of 3 stages.

    1) Preliminary retrieval of top-500 documents with TF-IDF

    2) Supporting document retrieval and supporting sentence prediction

    3) Answer Extraction using BERT large

  - The negative hop sequence are constructed by traversing through entities in Wikipedia.

HopRetriever : Retrieve Hops over Wikipedia to Answer Complex Questions [AAAI 2021]

- Experiments
  - HopRetriever works more effectively on the bridging questions.
  - The ground-truth supporting documents of the bridging questions are stringed with mentions that can provide informative structured knowledge, so HopRetriever performs better by leveraging mentions additionally.

Model	**Ans exists**			**Sent exists**			**All docs exist**		
Recall @	top-1	top-5	top-8	top-1	top-5	top-8	top-1	top-5	top-8
PathRetriever (Comparison)	77.00	**81.17**	82.25	88.33	90.36	90.62	**86.42**	**89.58**	**90.38**
HopRetriever (Comparison)	**77.40**	80.97	**82.31**	**91.31**	**92.73**	**92.79**	84.26	85.41	85.41
PathRetriever (Bridging)	81.95	91.08	91.92	80.56	88.29	89.20	70.77	85.25	86.63
HopRetriever (Bridging)	**89.27**	**93.66**	**94.19**	**87.73**	**92.79**	**93.29**	**82.11**	**89.41**	**90.01**

Table 3: Evidence collection results on different types of questions.

	Model	**Ans**		**Sup**		**Joint**	
		EM	F1	EM	F1	EM	F1
dev	Cognitive Graph QA (Ding et al. 2019)	37.55	49.40	23.11	58.52	12.18	35.28
	Semantic Retrieval (Nie et al. 2019)	46.41	58.70	39.86	71.53	26.53	49.00
	PathRetriever (Asai et al. 2020)	60.49	73.30	49.16	76.05	35.82	61.43
	HopRetriever	**62.07**	**75.18**	**52.53**	**78.92**	**37.81**	**64.50**
	HopRetriever-plus	66.56	79.21	56.02	81.81	42.01	68.97
test	DecompRC (Min et al. 2019)	30.00	40.65	-	-	-	-
	Cognitive Graph QA (Ding et al. 2019)	37.12	48.87	22.82	57.69	12.42	34.92
	DrKIT (Dhingra et al. 2020)	42.13	51.72	37.05	59.84	24.69	42.8
	Semantic Retrieval (Nie et al. 2019)	45.32	57.34	38.67	70.83	25.14	47.60
	Transformer-XH (Zhao et al. 2019)	51.60	64.07	40.91	71.42	26.14	51.29
	PathRetriever (Asai et al. 2020)	60.04	72.96	49.08	76.41	35.35	61.18
	Semantic Retrieval + HGN (Fang et al. 2019)	59.74	71.41	51.03	77.37	37.92	62.26
	HopRetriever	**60.83**	**73.93**	**53.07**	**79.26**	**38.00**	**63.91**
	HopRetriever-plus	64.83	77.81	56.08	81.79	40.95	67.75

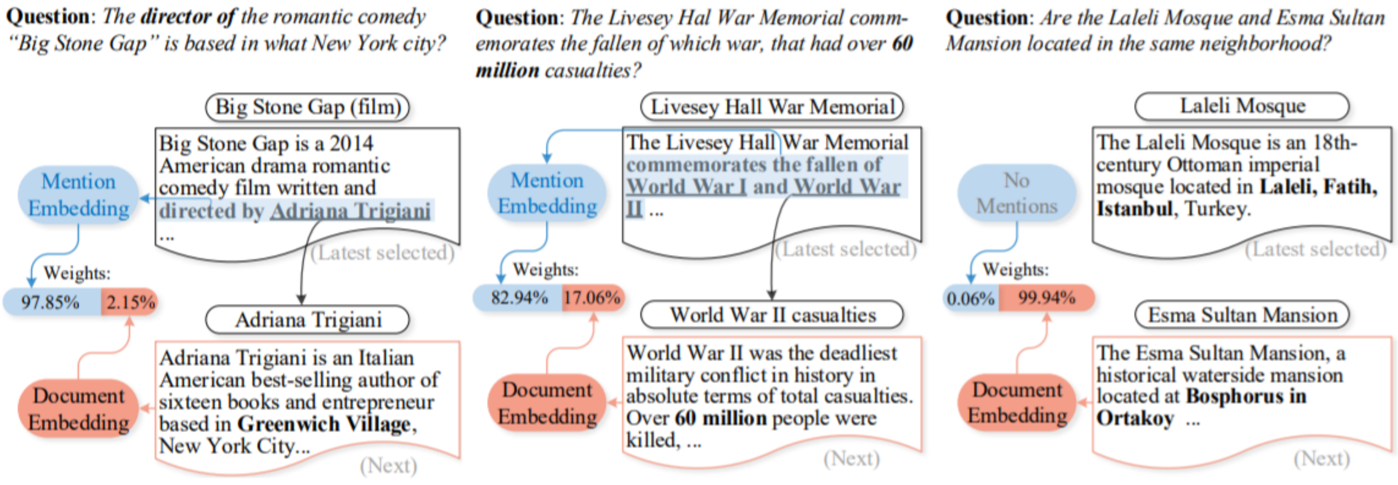
Table 4: Answer extraction and supporting sentence prediction result in the fullwiki setting of HotpotQA.

HopRetriever : Retrieve Hops over Wikipedia to Answer Complex Questions [AAAI 2021]

- Experiments
  - The authors analyze the weights and find that learnable weights provide intuitive explanation about <u>which embedding is more important for different question types</u>.

Question Type	Mention	Document
Bridging	89.53%	10.47%
Comparison	4.61%	95.39%

Table 5: Weights of mention embedding and document embedding on bridging questions and comparison questions.



(a) Case 1. (b) Case 2. (c) Case 3.

Figure 5: The weights of mention embedding and document embedding in different cases.

HopRetriever : Retrieve Hops over Wikipedia to Answer Complex Questions [AAAI 2021]

- Ablation Studies
  - **Structured knowledge within Wikipedia** is important for multi-hop evidence retrieval.
  - **Weighting between structured and unstructured knowledge** is important.

Model	**Ans exists**			**Sent exists**			**All docs exist**		
Recall @	top-1	top-5	top-8	top-1	top-5	top-8	top-1	top-5	top-8
full	86.89	91.11	91.80	88.41	92.78	93.20	82.54	88.60	89.09
1. w/o structured knowledge	76.35	86.02	88.12	80.91	88.49	89.92	66.20	78.89	81.23
2. w/o weighting	86.21	91.07	91.52	87.73	92.55	93.09	81.38	88.09	88.70
3. w/o sentence prediction	86.58	90.88	91.51	87.98	92.54	92.98	82.03	88.29	88.89

Table 6: Ablation experiments of HopRetriever.

Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering

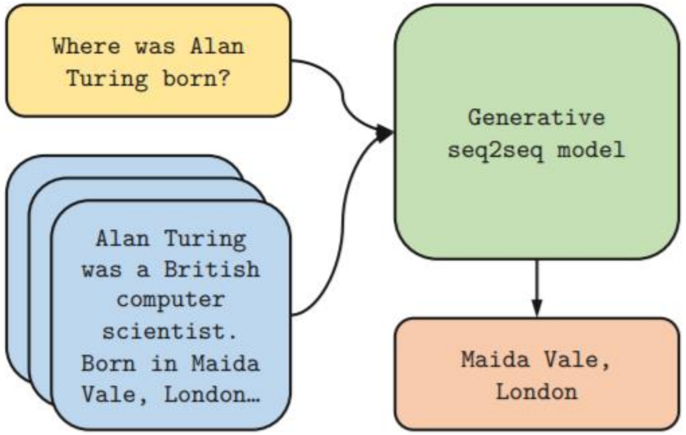
Gautier Izacard[1,2,3] **Edouard Grave**[1]
[1] Facebook AI Research, Paris
[2] ENS, PSL University, Paris
[3] Inria, Paris
gizacard|egrave@fb.com

EACL 2021

Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering [EACL 2021]

- Recent work have shown that factual information can be extracted from large scale language models trained on vast quantities of data.
  - *Radford et al., 2019; Petroni et al., 2019; Jiang et al., 2019; Talmor et al., 2019*
  - However, it **requires models containing billions of parameters**, since all the information needs to be stored in the weights.
  - The authors investigate how much this method could benefit from **having access to an external source of knowledge**, such as Wikipedia.

  - More specifically, the authors explore on building exciting developments in **generative modeling** and **retrieval** for open domain QA.
  - The authors say that this is evidence that <u>generative models are good at combining evidence</u> from multiple passages compared to extractive ones.



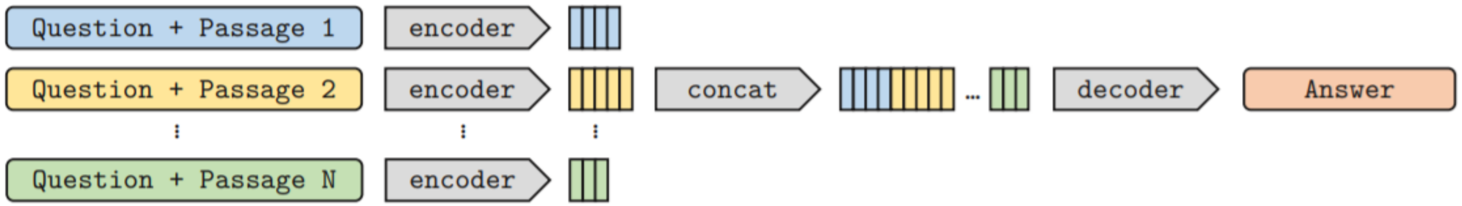


Figure 2: Architecture of the Fusion-in-Decoder method.

Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering [EACL 2021]

- Methodology
  - For retrieval of supporting passages, the authors consider two methods : **BM25** and **DPR**.
  - **Retrieval** is performed using approximate nearest neighbors with **FAISS library**.

  - The **Reader** is a seq2seq network such as **T5** or **BART**.
  - Each retrieved passage and its title are concatenated with the question.
  - The model performs evidence fusion in the decoder only, so the authors refer it as **Fusion-in-Decoder**.

  - By processing passages independently in the encoder, but jointly in the decoder, this method differs from Min et al., (2020); and Lewis et al., (2020).
  - Processing passages independently in the encoder allows to scale to large number of contexts, as it only performs self attention over one context at a time. => O(N)
  - On the other hand, processing jointly in the decoder allows to better aggregate evidence from multiple passages.

Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering [EACL 2021]

- Experiments
  - The authors test FiD with Natural Questions, TriviaQA and SQuAD v1.1.
  - The authors use base(220M) and large(770M) version of T5.
  - For both training and testing, the authors retrieve 100 passages and truncate them to 250 word pieces.
  - Passages are retrieved with DPR for NQ and TriviaQA and with BM25 for SQuAD.
  - The answers are generated using greedy decoding.

Model	NQ	TriviaQA		SQuAD Open	
	EM	EM	EM	EM	F1
DrQA (Chen et al., 2017)	-	-	-	29.8	-
Multi-Passage BERT (Wang et al., 2019)	-	-	-	53.0	60.9
Path Retriever (Asai et al., 2020)	31.7	-	-	**56.5**	**63.8**
Graph Retriever (Min et al., 2019b)	34.7	55.8	-	-	-
Hard EM (Min et al., 2019a)	28.8	50.9	-	-	-
ORQA (Lee et al., 2019)	31.3	45.1	-	20.2	-
REALM (Guu et al., 2020)	40.4	-	-	-	-
DPR (Karpukhin et al., 2020)	41.5	57.9	-	36.7	-
SpanSeqGen (Min et al., 2020)	42.5	-	-	-	-
RAG (Lewis et al., 2020)	44.5	56.1	68.0	-	-
T5 (Roberts et al., 2020)	36.6	-	60.5	-	-
GPT-3 few shot (Brown et al., 2020)	29.9	-	71.2	-	-
Fusion-in-Decoder (base)	48.2	65.0	77.1	53.4	60.6
Fusion-in-Decoder (large)	**51.4**	**67.6**	**80.1**	**56.7**	63.2

Table 1: Comparison to state-of-the-art. On TriviaQA, we report results on the open domain test set (left), and on the hidden test set (right), competitions.codalab.org/competitions/17208#results).

Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering [EACL 2021]

- Experiments
  - The authors observe that increasing the number of passages from 10 to 100 leads to 6% improvement on NQ.
  - On the other hand, the performance of most extractive models seems to peak around 10~20 passages.
  - Seq2Seq models are good at combining information from multiple passages.

	NaturalQuestions		TriviaQA	
Training Passages	w/o finetuning	w/ finetuning	w/o finetuning	w/ finetuning
5	37.8	45.0	58.1	64.2
10	42.3	45.3	61.1	63.6
25	45.3	46.0	63.2	64.2
50	45.7	46.0	64.2	64.3
100	46.5	-	64.7	-

Table 2: Performance depending on the number of passages used during training. Exact Match scores are reported on dev sets.

ANY QUESTIONS?