Few-shot Reranking for Multi-hop QA via Language Model Prompting

Muhammad Khalifa[*], **Lajanugen Logeswaran**[†], **Moontae Lee**[†‡],
Honglak Lee[*†], **Lu Wang**[*]
University of Michigan[*], LG AI Research[†], University of Illinois at Chicago[‡]

Abstract

We study few-shot reranking for multi-hop QA (MQA) with open-domain questions. To alleviate the need for a large number of labeled question-document pairs for retriever training, we propose **PROMPTRANK**, which relies on language model prompting for multi-hop path reranking. PROMPTRANK first constructs an instruction-based prompt that includes a candidate document path and then computes the relevance score between a given question and the path based on the conditional likelihood of the question given the path prompt according to a language model. PROMPTRANK yields strong retrieval performance on HotpotQA with only 128 training examples compared to state-of-the-art methods trained on thousands of examples — 73.6 recall@10 by PROMPTRANK vs. 77.8 by PathRetriever (Asai et al., 2020) and 77.5 by multi-hop dense retrieval (Xiong et al., 2021).[1]

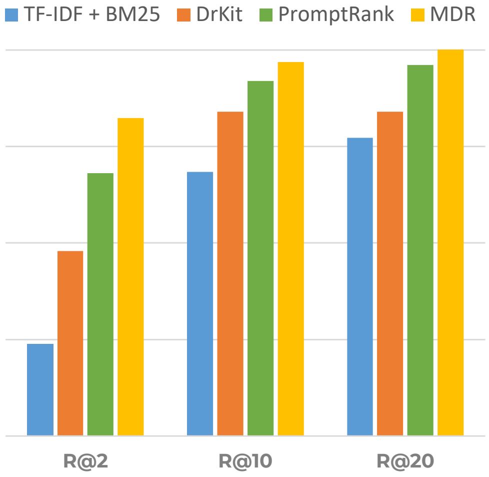


Figure 1: Multi-hop retrieval recall@k on HotpotQA. PROMPTRANK, using only 128 examples, outperforms DrKit (Dhingra et al., 2020) and performs closely to Multi-hop Dense Retrieval (Xiong et al., 2021). Both fully supervised models are trained on ~90K examples.

1 Introduction

Many information-seeking queries are in the form of multi-hop questions. For instance, to answer the question *"What 1988 Christmas comedy film did Brian-Doyle Murray star in?"*, we need to **(i)** search for movies starring Brian Murray, then **(ii)** identify which of them were released in 1988 during Christmas. Evidence required to answer such questions is often dispersed in different documents, requiring sequential, multi-step reasoning to reach the answer (Perez et al., 2020), typically referred to as multi-hop question answering (MQA).

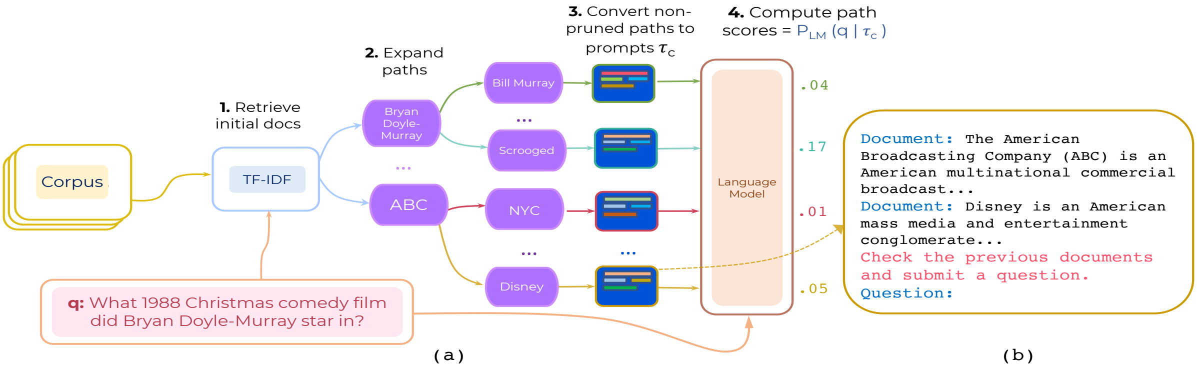
Given a multi-hop question and a large document corpus, existing MQA systems largely follow a *retrieve-then-read* pipeline, where a retriever module first identifies relevant documents from the corpus, and a reader module produces the answer based on the retrieved output (Asai et al., 2020; Li et al., 2021; Singh et al., 2021; Qi et al., 2021). The retriever module is trained to predict the ground-truth evidence document(s) given the question (Karpukhin et al., 2020; Qi et al., 2021;). However, curating large datasets of question-document pairs is expensive, especially for low-resource languages or domains that require unique expertise (e.g., medical or legal documents), thus creating a *bottleneck* for building QA pipelines (Ram et al., 2022). Moreover, resorting to heuristics for data labeling can lead to incorrect annotation (Izacard and Grave, 2021). This difficulty is further exacerbated in the case of multi-hop questions, as they need to be annotated with multiple support documents.

The majority of existing data-efficient retrieval and reranking methods are restricted to *single-hop* QA, and it is unclear how to extend them to the *multi-hop* setting. For instance, Ram et al. (2022) proposed "recurrent span retrieval" to obtain *psuedo* question-document pairs in an unsupervised way for single-hop QA. However, in the multi-hop

[*] Correspondence to `khalifam@umich.edu`

[1] Code available at `https://github.com/mukhal/PromptRank`.

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 15882–15897
July 9-14, 2023 ©2023 Association for Computational Linguistics


1. Retrieve initial docs
2. Expand paths
3. Convert non-pruned paths to prompts τ_c
4. Compute path scores = $P_{LM}(q \mid \tau_c)$
Corpus
TF-IDF
Bryan Doyle-Murray
ABC
Bill Murray
Scrooged
NYC
Disney
Language Model
.04
.17
.01
.05
q: What 1988 Christmas comedy film did Bryan Doyle-Murray star in?
(a)
(b)

```
Document: The American
Broadcasting Company (ABC) is an
American multinational commercial
broadcast...
Document: Disney is an American
mass media and entertainment
conglomerate...
Check the previous documents
and submit a question.
Question:
```

Figure 2: An overview of the full retrieval system. (a): Initial documents from TF-IDF are retrieved and expanded based on hyperlinks for H times. PromptRank converts each path into a prompt τ_c and scores through $P_{\text{LM}}(q|\tau_c)$ for a given question q using a language model. For simplicity, we omit intermediate scoring steps where paths of length $h < H$ are scored using the same fashion and only the top-scored ones are expanded. (b): A sample of what a 2-hop path prompt looks like. Prompts are constructed in terms of an **instruction** and the **document path**.

case, it is less likely that we can retrieve recurrent spans from multiple documents that follow a valid reasoning trajectory. Moreover, their method requires intensive pretraining on the obtained corpus. Seonwoo et al. (2021) focus on weakly supervised multi-hop QA retrieval, yet their method uses corpus-specific (e.g., Wikipedia) heuristics and also requires pretraining. This motivates the need for data-efficient multi-hop retrieval methods that **(i)** work out-of-the-box without requiring additional (pre)training, and **(ii)** do not rely on hand-designed heuristics for data collection and annotation.

To this end, we present **PromptRank**, which leverages the power of large language models (LLMs) for few-shot multi-hop retrieval. PromptRank combines a simple unsupervised retrieval method i.e., TF-IDF similarity, with an LLM reranker that scores the relevance of document paths to a question based on the *conditional likelihood of generating the question given the path*. Our approach makes use of instruction-based prompting (Sanh et al., 2021; Ouyang et al., 2022) to steer the LLM towards assigning higher scores to more relevant support document chains.[2] To calibrate the model's reranking scores and alleviate prompt sensitivity (Zhao et al., 2021b), we borrow techniques from the literature such as *temperature scaling* (Kull et al., 2019) and *instruction ensembling* (Schick and Schütze, 2021a). We also utilize *demonstration ensembling* to leverage more examples than what can fit into the context of transformer LLMs by combining reranking probabilities computed with different demonstrations.

We evaluate few-shot PromptRank on HotpotQA (Yang et al., 2018), a standard MQA benchmark, and show that it compares favorably against state-of-the-art models while using orders of magnitude fewer examples. More precisely, with only 128 training examples, PromptRank outperforms DrKit (Dhingra et al., 2020) and is only 4.2 Recall@10 points lower than multi-hop dense retrieval (MDR) (Xiong et al., 2021) (see Figure 1). We also showcase PromptRank as part of a QA pipeline, again, displaying close QA performance to fully-supervised retrievers—only 4.1 F_1 points lower than MDR.

In summary, our contributions in this paper are:

1. We propose PromptRank, a few-shot reranking approach for multi-hop QA that reranks a given document path based on the likelihood of generating the question given a path prompt.
2. PromptRank exhibits strong few-shot retrieval performance with as few as 128 examples and compares favorably to fully supervised methods (§3.1).
3. PromptRank leads to strong QA performance when combined with a pretrained reader module, performing close to fully-supervised retrievers (§3.2).

2 PromptRank

An overview of the full retrieval system is displayed in Figure 2: Given a question q, the system expands sequences of supporting documents into paths of length H, which are used to answer the question. At each step, we first use TF-IDF similarity to ob-

[2] We use *path* and *chain* interchangeably throughout the paper.

tain an initial set of supporting document paths.[3] We then use PromptRank to rerank the current document chains based on their relevance to the question (§2.1).

Concretely, we start with retrieving F candidate documents using TF-IDF for the 'first hop'. These '1-hop' candidates are scored by PromptRank and K_1 top-ranked documents are kept and further expanded based on their hyperlinks to obtain 2-hop reasoning paths.[4] These 2-hop reasoning chains are again reranked and the most promising K_2 candidates are further expanded. The process repeats until we obtain paths of length H, where H can be a hyperparameter.[5] As the document graph can have a high branching factor, we only keep the top-L hyperlinks as reranking candidates based on TF-IDF similarity between the hyperlink document and the question. We have found this pruning step to improve efficiency without much performance drop. This process is shown in Figure 2(a).

2.1 Path Reranking with PromptRank

Given a question q and a reasoning path or chain c, we use an LM to score c according to its relevance to q. Concretely, we measure *the likelihood of the question* given the path as follows:

$$\text{Score}_q(c) = P_{\text{LM}}(q|\tau_c) \qquad (1)$$

where $P_{\text{LM}}(q|\tau_c)$ is the conditional probability of generating the question given a prompt τ_c containing path τ_c using an LM. Our initial experiments show that using $P_{\text{LM}}(q|\tau_c)$ works *substantially* better than $P_{\text{LM}}(c|\tau_q)$ for a question-containing prompt τ_q, which agrees with the findings in dos Santos et al. (2020).[6] We argue that two factors contribute to this gap. First, LMs can be sensitive to the *surface form* (Holtzman et al., 2021) of reasoning paths, making it difficult to reliably compare the probabilities of different reasoning paths using $P_{\text{LM}}(c|\tau_q)$. For instance, $P_{\text{LM}}(c|\tau_q)$ tends to be higher for shorter paths. On the other hand, $P_{\text{LM}}(q|\tau_c)$ does not suffer from this issue since we compare the probabilities of the same string (i.e., the question) by conditioning on different reasoning paths. Second, the prompt format using $P_{\text{LM}}(q|\tau_c)$—the question follows a document—agrees more with the web data used for LM pretraining, where documents are usually followed by FAQs, questionnaires, and surveys, rather than the other way around. We further add a *temperature parameter* to scale the model output logits before computing $P(q|\tau_c)$. This can be seen as an instance of model calibration (Guo et al., 2017; Desai and Durrett, 2020; Jiang et al., 2021) with the goal of improving the reranking scores. We show that temperature scaling boosts reranking performance in §3.1.

Constructing Prompt τ_c As shown in Figure 2 (b), the prompt consists of an instruction along with the document path. The instruction's goal is to encourage higher scores for more relevant paths by eliciting the LM reasoning ability (Ouyang et al., 2022). We note that the instruction part is *fixed* across all prompts constructed for different candidate paths.

The path is expressed in the prompt by *concatenating* all documents in the chain and prepending each document with a fixed prefix, such as *"Document:"* or *"Passage:"*. The concatenation of path documents significantly improves reranking by *simultaneously* considering all hops, which allows the LM to do a context-aware evaluation of path relevance.

2.2 Instruction Search and Ensembling

Although instructions can be manually engineered to trigger the LM to accomplish the task (e.g., *"Read the following documents and generate a question"*), this requires human expertise and can be sub-optimal. Therefore, we leverage automated instruction search Gao et al. (2021), where we use an encoder-decoder LM, e.g., a T5-Base model (Raffel et al., 2020), that is trained to fill masked text spans to generate instructions.

Specifically, we fill in the template *"Task: <X> documents and <Y> question. Question:"*, where <X> and <Y> are the masked spans expected to be filled in by the model (e.g., for a human-written instruction example, <X> = "*Read the following*" and <Y> = "*answer the*"). We consider two variations of this template corresponding to the cases where the document path appears before/after the template. We constrained the template to contain the words 'documents' and 'question' to ensure that the model generates relevant prompts. We have

[3] PromptRank is agnostic to the retrieval approach and can be combined with other retrieval techniques.

[4] We assume the presence of hyperlinks following previous work (Asai et al., 2020; Qi et al., 2021) although PromptRank is agnostic to how a candidate path is obtained.

[5] This process can be viewed as a variation of beam search.

[6] Earlier experiments showed that the recall of $P_{\text{LM}}(q|\tau_c)$ was at least 60% better than that of $P_{\text{LM}}(c|\tau_q)$.

found that using a less specific template without such tokens leads to more diverse but less relevant instructions. The exact templates used are in Appendix A.2.

Previous work has shown that mixing multiple prompts can improve few-shot performance (Gao et al., 2021; Schick and Schütze, 2021b). Similarly, such ensembling could produce more regularized path scores by alleviating prompt sensitivity (Zhao et al., 2021b). Given a path, we combine the scores of obtained through different instructions. We experiment with both *mean* and *max* ensembling.

2.3 Demonstration Ensembling

We employ in-context learning (ICL) (Brown et al., 2020) to teach the LLM to do reranking by showing the model examples i.e., demonstrations of questions and their gold paths. A major obstacle to this approach is the *input length limit* in standard transformer LMs. Since paths are comprised of multiple documents, in most cases we cannot feed more than two demonstrations without exceeding the limit of 1024 tokens, a standard setup for pretrained LMs. To workaround that, we utilize *demonstration ensembling*, where different in-context demonstrations are used to compute scores for a given path, and the scores are combined by a mean or max operation.

3 Experiments

Data We evaluate our method on **HotpotQA** (Yang et al., 2018), which consists of two-hop questions over diverse topics. We focus on the *fullwiki* setting in which two Wikipedia passages are required to answer the questions. Since the gold passages for the test set are not available, we follow prior work and evaluate PromptRank on the development set, which has 7,405 questions. There are two main question types in HotpotQA: (1) *comparison* questions usually require contrasting two entities and (2) *bridge* questions can be answered by following a connecting entity that links one document to another. We also evaluate few-shot PromptRank on the **2WikiMQA** dataset (Ho et al., 2020) in Appendix D.1.

Compute Infrastructure All our reranking experiments are run on a workstation with a single Nvidia A40 GPU and 256GB of RAM. Our QA experiments in §3.2 are run on a workstation with two Nvidia Quadro RTX 8000 GPUs and 128GB of RAM.

Models We use HuggingFace implementations (Wolf et al., 2020) of GPT2-XL (1.5B) (Brown et al., 2020), T5-Base (220M), T5-Large (770M) and T5-XL (3B) (Raffel et al., 2020) in our experiments. We use the 'LM adapted' version of T5 models since they have been shown to work better for prompt-based learning (Lester et al., 2021). We report additional results with the OPT-30B model (Zhang et al., 2022) in §4.3.

Hyperparameters For PromptRank, we use a path length of $H = 2$ for all experiments. For pruning the search space we use $K_1 = 5$ and $L = 3$. We use the TF-IDF index implemented by Asai et al. (2020) and initially retrieved $F = 100$ documents from TF-IDF. We truncate path documents to 230 tokens before constructing the prompt and limit the prompt length to 600 tokens. When using in-context demos, we use the maximum length of 1024 tokens.

Metrics Retrieval performance is measured using both Recall (R@k) and Answer Recall (AR@k), with $k \in \{2, 10, 20\}$. R@k measures whether the two gold documents are present in the top-k retrieved documents and AR@k is the recall of the answer string in the top-k retrieved documents. For HotpotQA, we only compute AR over questions with span answers (we ignore yes/no and comparison questions). Since we do not have access to the HotpotQA test set, we report results on the original development set provided by Yang et al. (2018).

Document Scores We compute document scores from path scores as follows. Similar to Das et al. (2019), we take a document score to be the *maximum* of all its path scores. We find this change to yield better recall than using path scores, with details elaborated in Appendix B.

Instruction Search and Temperature For instruction search, we generate 200 different instructions as described in §2.2 using top-k sampling with $k = 10$. Then, we select the best instruction based on R@2 over our development set of 128 examples. The same process is used to select the optimal temperature value. Table A1 shows the best 10 instructions identified.

Baselines We compare our reranker to the following baselines. **TF-IDF** retrieves top similar documents to the question using TF-IDF similarity and **TF-IDF + BM25** adds an extra step where retrieved documents and their hyperlinks are reranked using

	# Ex.	R@2	R@10	R@20	AR@2	AR@10	AR@20
Unsupervised Baselines							
TF-IDF	–	9.9	27.6	35.0	37.6	53.8	60.2
TF-IDF + BM25	–	19.1	54.7	61.8	49.5	74.7	79.9
Fully-supervised Baselines							
DrKit	~90K	38.3	67.2	71.0	–	–	–
MDR	~90K	65.9	77.5	80.2	–	–	–
PathRetriever	~90K	66.4	77.8	78.7	82.2	90.5	90.5
PROMPTRANK, *no ICL*							
GPT2-XL†	–	36.6	60.5	65.9	63.0	83.9	87.4
T5-XL†	–	42.8	68.9	74.1	69.3	86.8	89.0
+ best inst.	128	47.8	71.4	76.0	74.0	87.9	89.7
+ temp. scaling	128	49.7	71.9	76.2	76.2	88.4	89.9
+ inst. ensemble	128	51.3	72.0	76.4	77.6	88.5	90.3
PROMPTRANK, *with ICL*							
T5-XL, $N_{\text{demos}} = 2$	128	52.3 (.7)	73.1 (.2)	**77.1** (.2)	78.6 (.7)	88.7 (.0)	90.3 (.1)
T5-XL, $N_{\text{demos}} = 8$	128	**54.5** (.7)	**73.6** (.3)	76.9 (.1)	**79.1** (.6)	**89.0** (.1)	**90.5** (.0)
T5-XL, $N_{\text{demos}} = 10$	128	54.4 (.5)	73.5 (.3)	76.9 (.1)	78.9 (.4)	88.9 (.1)	**90.5** (.0)

Table 1: Retrieval performance on HotpotQA comparing PROMPTRANK to baselines. †: No instruction used. PROMPTRANK results except those marked with † use a labeled set of 128 examples for tuning the instruction and the temperature parameter. Few-shot experiments use the best instruction found on a held-out set of 128 examples (See Table A1 in Appendix) and temperature ($T = 1.4$). In-context learning (ICL) experiments are run 5 times with demos sampled from the same 128-example set and we report mean and (std). Our best results are highlighted in **bold**.

BM25 (Robertson et al., 1995). **PathRetriever** (Asai et al., 2020) is a graph-based retriever trained to expand an initial pool of documents based on Wikipedia links and searches for the best path using beam search.[7] **DrKIT** (Dhingra et al., 2020) is an end-to-end trained dense retrieval approach that starts from question entities and traverses a virtual knowledge base to find the relevant entities. Multi-hop Dense Retrieval **(MDR)** (Xiong et al., 2021) encodes the question and the documents retrieved by each step into a dense vector and uses maximum inner-product search (MIPS) to find the next hop.

Below, we start with the evaluation of the zero- and few-shot reranking of PROMPTRANK (§3.1). Then, we move to evaluate downstream MQA performance in the few-shot setting (§3.2).

3.1 Retrieval Performance

Table 1 shows the performance of PROMPTRANK and other comparisons in zero- and few-shot settings.

Zero-shot Performance We start with discussing the retrieval performance of zero-shot PROMPTRANK on HotpotQA. First, we observe that simple TF-IDF performs poorly in terms of different recall metrics, while TF-IDF + BM25 performs much better, yet still worse than fully-supervised approaches. Next, we look at the performance of the zero-shot PROMPTRANK (T5-XL) which uses no instructions, i.e., the prompt consists of only the document path. These models obtain better recalls than TF-IDF + BM25 and even outperform the fully-supervised DrKit. Although this approach does not use *any* labeled data, it is only 3.7 AR@10 points worse than PathRetriever, which is trained on ~90K examples. These findings demonstrate PROMPTRANK's effectiveness at reranking paths of documents.

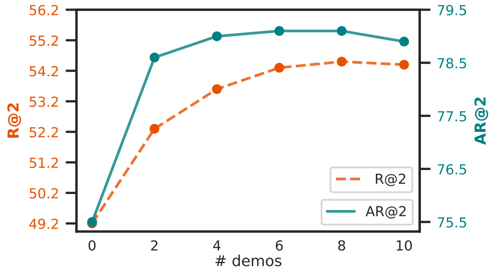


Figure 3: Demonstration ensembling (§2.3) is able to leverage more examples ($N > 2$) than what is allowed by the T5-XL context window size. We show R@2 and AR@2 on HotpotQA with different numbers of demonstrations. Metrics are averaged over 5 runs with different demos sampled from a 128-example set.

Few-shot Performance The zero-shot performance of PROMPTRANK can be further improved with access to a small set of labeled examples (in

[7] We run PathRetriever on HotpotQA with original hyperparameters except for an initial TF-IDF pool size=100 to allow for fair comparison to our approach.

our case, we only used 128 examples from HotpotQA) for instruction search and finding temperature value. We observe a substantial boost of 11.6% (42.8 → 47.8) in R@2 of PromptRank when using the *best instruction* found by instruction search. Furthermore, temperature scaling with $T = 1.4$ also provides a boost of 3.9% (47.8 → 49.7) points in R@2.

We also observe that instruction ensembling gives a further performance boost, reaching 51.3 R@2 with PromptRank. We show the performance of *max ensembling*, which we have found to perform better than mean ensembling in terms of R@2. We hypothesize that max ensembling computes an *upper bound* on the path scores, compensating for any underestimation of path scores that can happen when using a single instruction.

In-context learning We experiment with an N-shot setting while making sure that the two demonstrations cover *both* question types in HotpotQA (bridge and comparison). Figure 3 shows that both R@2 and AR@2 improve as we use more demonstrations. With only 2 examples, we observe a large boost of 6.3% (49.2 → 52.3) in R@2. Since we cannot fit more than 2 demonstrations in the 1024 context window, we use demonstration ensembling (§2.3). For instance, 6-shot ensembling scores a path by combining 3 different contexts, each obtained using 2 demonstrations. We use max ensembling as it is found to work best. Figure 3 shows the in-context learning performance with a different number of demonstrations. We observe a steady increase in R@2 until $N = 8$. AR@2 also improves with more demonstrations but drops slightly with $N = 10$. Interestingly, demonstration ensembling has enabled us to leverage more examples than permitted by the context window size of T5-XL. We leave it to future work to study the applicability of this technique to other tasks.

3.2 Full QA Performance

We analyze the performance of PromptRank when used as the retriever in a QA pipeline. We adopt an extractive reader model based on ELECTRA Large (Clark et al., 2020) with two heads to predict the start and end of the answer span. We use the checkpoint provided by Xiong et al. (2021), and the same inference setting. Details on the inference hyperparameters for the reader are in Appendix C.1.

Retriever	EM	F1
Fully-supervised		
MDR (Xiong et al., 2021)	62.3	75.1
Zero-shot		
TF-IDF	39.6	49.4
PromptRank, no inst	55.7	67.7
Few-shot		
PromptRank, ($N_{\text{demos}} = 2$)	57.8 (.1)	70.0 (.1)
PromptRank, ($N_{\text{demos}} = 10$)	58.3 (.0)	70.5 (.1)

Table 2: Answer EM and F1 on HotpotQA development set. PromptRank results are aggregated over 3 runs with different demonstrations. We show metrics mean and (std). To allow for a fair comparison, only the retriever is varied over these systems while the reader module is the *same*.

Retriever	EM	F1
DrKit (Dhingra et al., 2020)	42.1	51.7
PathRetriever (Asai et al., 2020)	60.0	73.0
MDR (Xiong et al., 2021)	62.3	75.3
PromptRank, ($N_{\text{demos}} = 2$)	58.1	71.1

Table 3: Answer EM and F1 on HotpotQA test set. MDR and PromptRank use the same ELECTRA reader, while other systems use different readers.

In Table 2, we compare the QA performance on HotpotQA **development set** with PromptRank as the retriever against a *fully-supervised* retriever, namely MDR (Xiong et al., 2021) as well as unsupervised TF-IDF. PromptRank with $N_{\text{demos}} = 10$ is only 4.6 F$_1$ points worse than MDR, which is using the *same* reader module. Table 3 shows performance on HotpotQA **test set** with different *fully-supervised* systems compared to PromptRank ($N_{\text{demos}} = 2$), where PromptRank is only 1.9 and 4.2 EM points below PathRetriever and MDR, respectively.

4 Analysis

4.1 Comparison to Single-hop Reranking

The key idea behind our approach is to conduct joint reasoning with documents in the path using the LM, as opposed to reranking each document in the path separately (*single-hop reranking*). More specifically, in single-hop reranking, we expand paths using the same setup of PromptRank but rerank each document d separately using $p(q|\tau_d)$, for a given document prompt τ_d.

To assess whether our multi-hop reranking approach offers the advantage of global reasoning, we compare both approaches by running two experiments with identical settings except for how documents are reranked. For evaluation, we use a set of 4K questions from HotpotQA and T5-Large, and

Re-ranking	R@2	R@10	AR@2	AR@10
Single-hop	22.8	52.0	54.9	73.8
Multi-hop	46.9	67.6	75.4	87.9

Table 4: Recall measured on 4K questions from HotpotQA in two settings: reranking each document separately with the LM (single-hop) and reranking the full path at once (multi-hop). Multi-hop reranking performs substantially better than single-hop.

no instruction is used, i.e., the prompt only contains the document(s). Table 4 shows the retrieval performance of both approaches. Interestingly, a large gap in recall scores is observed between single-hop and multi-hop reranking. This supports our hypothesis that jointly considering multiple documents in the path helps the LM better model documents' relevance to the question.

4.2 Role of Instruction

Our goal here is to investigate **(i)** how useful is the presence of the instruction in the prompt, **(ii)** how much benefit (if any) automated instruction search provides over manual instructions, and **(iii)** whether the instruction's location in the prompt matters. To answer these questions, we analyze the recall over 200 different instructions generated using the method described in §2.2 and using 1K examples from HotpotQA with different LM sizes: T5-XL, T5-Large, and T5-Base, with results displayed in Figure 4. This analysis uses an initial set of TFIDF documents of size $F = 30$.

Usefulness of Instruction We can see that using no instruction consistently yields poorer performance than using an instruction of any sort, across all variants of T5. Interestingly, without the instruction, the three model sizes have almost the same R@2. The difference in their performances becomes apparent when an instruction is added. Strikingly, in the no instruction case, T5-Large performs *worse* than T5-Base in terms of AR@2, showing that scaling does not consistently help recall when no instructions are used. This hints at the fact that instructions play a major role in harnessing the full power of LLMs, at least for our task.

Benefit of Automated Instruction Search Next, we compare a human-written instruction against an instruction found through automated instruction search on a labeled set of 128 examples. The manual instruction we use is *"Please write a question based on these passages."*, which is used by Sachan et al. (2022).[8] In Figure 4, we compare the recall when using these instructions. Interestingly, the search-based instruction outperforms the manual one in almost all cases. We also observe that the manual instruction performs poorly for AR@2 on T5-base, even worse than no instruction. These observations hint at the utility of automated instruction search for path reranking. However, it is worth noting that the best instruction on a relatively small held-out set will not necessarily generalize during test time: The search-based instruction produces AR@2 and R@2 that are almost the same or worse than the median instruction, respectively with T5-Large.

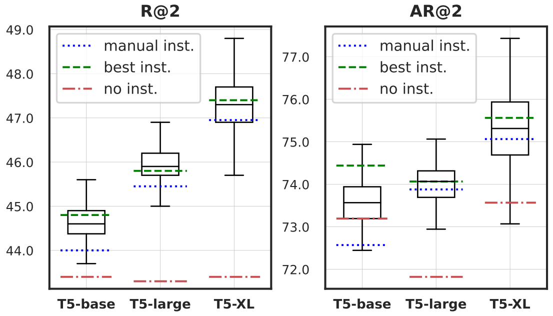


Figure 4: R@2 and AR@2 with different kinds of instructions for three different T5 sizes: XL, Large, and Base. The recall is measured over 1K questions from HotpotQA train set using 200 different instructions obtained using automated search §2.2.

Location of Instruction We study the performance of two different kinds of prompts, where the instruction appears *before* and *after* the path. Figure 5 shows the R@2 and AR@2 in both cases for T5 models of different sizes. We observe that placing the instruction after the path performs *consistently better* than placing before it, across all model variants. We hypothesize this to be an instance of the *recency bias* exhibited by LMs (Zhao et al., 2021b), *i.e.,* placing the instruction right before where the model is asked to generate the question better primes the LM for the task and produces better calibrated path scores. We expect such finding to generalize to other tasks where instruction-based prompting is used.

4.3 Choice of Language Model

Table 5 compares the reranking performance of GPT2-XL and OPT-30B (Zhang et al., 2022) mod-

[8] We average recall of the two cases where the instruction falls before and after the path. See the next paragraph for more context.

Model	R@2	R@10	AR@2	AR@10
OPT-30B	36.9	65.4	**61.0**	82.0
GPT2-XL	**47.2**	**70.3**	57.1	**85.7**

Table 5: Document and answer recall of GPT2 and OPT models based on 1000 questions from HotpotQA.

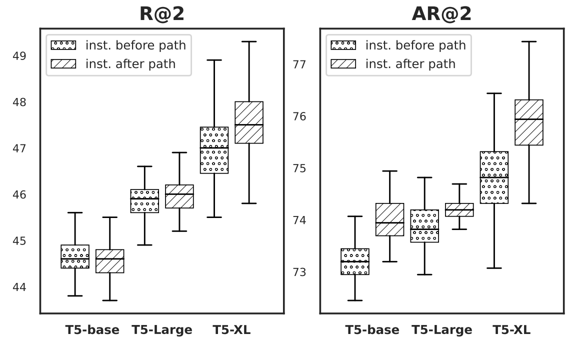


Figure 5: Retrieval performance when placing the instruction *before* and *after* the document path in the prompt. The recall is measured over 1K questions from HotpotQA train set using 200 different instructions. Having the instruction after the path performs consistently better which is likely due to recency bias (Zhao et al., 2021b).

els. Despite having an order of magnitude more parameters, we observe that the OPT model is generally worse compared to the smaller GPT2-XL model. We suspect this is due to domain mismatch between pre-training data and task relevant data. Pre-training data of GPT2 models is potentially more biased towards Wikipedia data compared to the OPT models which are trained on more diverse data. Importantly, this shows that scaling up the language model doesn't necessarily guarantee better reranking performance and domain gap is an important consideration.

4.4 Further Analysis and Comparisons

We further analyze the inference cost of PROMPTRANK compared to PathRetriever and MDR in Appendix D.2. In Appendix D.3, we study PROMPTRANK's recall sensitivity to document order in the prompt τ_c by comparing performance using two different document ordering schemes in the prompt. Lastly, we compare PROMPTRANK to few-shot PathRetriever and LOUVRE (Seonwoo et al., 2021) in Appendix D.4.

5 Related Work

Multi-hop Retrieval The majority of approaches for multi-hop question answering rely on two main components: a retriever and a reader. The retriever component can be a sparse index or heuristic-based such as TF-IDF or BM25 (Chen et al., 2017; Nie et al., 2019) or dense index (Karpukhin et al., 2020; Xiong et al., 2021; Zhao et al., 2021a). Other approaches aimed to improve the retriever with an additional *reranking* step on top of a simple retriever (Wang et al., 2018; Lee et al., 2018; Htut et al., 2018). Asai et al. (2020) combined TF-IDF retriever with a recurrent graph retriever and used the reader module to rerank paths based on answer confidence. Qi et al. (2021) used a single transformer model to perform retrieval, reranking and reading in an iterative fashion. However, the good performance of previous work comes mainly from training on a large number of examples and is likely to fail in low-data settings. To treat this issue, Seonwoo et al. (2021) proposed to pretrain MDR (Xiong et al., 2021) on a large number of weakly-supervised examples of questions and the corresponding document paths. Although promising in low-data settings, their pretraining is computationally expensive as it is done on millions of examples. On the other hand, our approach requires no task-specific pretraining.

Language Models Prompting Prompt-based learning aims to construct better inputs, *i.e.,* prompts to language models to elicit better zero- or few-shot performance (Brown et al., 2020; Liu et al., 2021). Recently, instruction tuning, where a language model is trained to follow natural language instruction, has shown impressive zero-shot performance on unseen tasks (Wei et al., 2021; Ouyang et al., 2022). In our work, we use instructions to guide to model toward assigning better scores to more relevant document paths.

LM-based Reranking Our scoring function is related to query likelihood retrieval (Lavrenko and Croft, 2017; Ponte and Croft, 2017) and is in line with previous work that employed generative language models for passage reranking (Nogueira et al., 2020). dos Santos et al. (2020) performed single-hop reranking using question likelihood given the passage, but their setting was limited to fully-supervised, single-hop QA. Concurrent with our work is (Sachan et al., 2022), where the authors leverage LLMs for unsupervised passage reranking for QA. While their focus is on single passages, we study the reranking of multi-passage paths, which is more challenging. Moreover, their exploration of prompting is limited to a single manual instruction, whereas we provide an in-depth analysis of the effect of different prompting aspects on the recall such as instruction importance, location in the prompt, and manual vs. automated.

Conclusion

We introduced PROMPTRANK, a method to perform few-shot reranking of multi-document paths for multi-hop question answering based on large language models. Experiments on a standard multi-hop QA benchmark show the strong performance of PROMPTRANK in the few-shot setting compared to fully-supervised multi-hop reranking systems. Future avenues of exploration include combining PROMPTRANK with efficient tuning techniques such as prefix tuning and efficient strategies for instruction search.

Limitations

One limitation to LM-based reranking is the computational overhead involved in reranking paths. Our approach requires a forward pass through the LM to rerank each path, which can become expensive when using relatively large models such as GPT-3 or when dealing with more hop count that creates combinatorially more paths. Another limitation of PROMPTRANK is imposed by the transformer context window length. Since PROMPTRANK requires the prompt to include all path documents, it could be infeasible to fit all path documents into the prompt for paths with a larger hop count. A potential direction to workaround this is to condense or summarize the path documents beforehand. We leave it to future work to explore this and other techniques.

Acknowledgements

This work is supported by LG AI Research. Additionally, we would like to thank Sewon Min for providing feedback on the paper. We also thank the anonymous reviewers for their valuable suggestions.

References

RNNLogic: Learning Logic Rules for Reasoning on Knowledge Graphs | OpenReview.

Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. 2020. Learning to retrieve reasoning paths over wikipedia graph for question answering. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

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, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1870–1879. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Rajarshi Das, Ameya Godbole, Manzil Zaheer, Shehzaad Dhuliawala, and Andrew McCallum. 2019. Chains-of-reasoning at textgraphs 2019 shared task: Reasoning over chains of facts for explainable multi-hop inference. In *Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-13)*, pages 101–117.

Shrey Desai and Greg Durrett. 2020. Calibration of pre-trained transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 295–302. Association for Computational Linguistics.

Bhuwan Dhingra, Manzil Zaheer, Vidhisha Balachandran, Graham Neubig, Ruslan Salakhutdinov, and William W. Cohen. 2020. Differentiable reasoning over a virtual knowledge base. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Cicero dos Santos, Xiaofei Ma, Ramesh Nallapati, Zhiheng Huang, and Bing Xiang. 2020. Beyond [cls] through ranking by generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1722–1727.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 3816–3830. Association for Computational Linguistics.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Ari Holtzman, Peter West, Vered Shwartz, Yejin Choi, and Luke Zettlemoyer. 2021. Surface form competition: Why the highest probability answer isn't always right. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7038–7051, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Phu Mon Htut, Samuel R. Bowman, and Kyunghyun Cho. 2018. Training a ranking function for open-domain question answering. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 2-4, 2018, Student Research Workshop*, pages 120–127. Association for Computational Linguistics.

Gautier Izacard and Edouard Grave. 2021. Distilling knowledge from reader to retriever for question answering. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. 2021. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9:962–977.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6769–6781. Association for Computational Linguistics.

Meelis Kull, Miquel Perello Nieto, Markus Kängsepp, Telmo Silva Filho, Hao Song, and Peter Flach. 2019. Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with dirichlet calibration. *Advances in neural information processing systems*, 32.

Victor Lavrenko and W Bruce Croft. 2017. Relevance-based language models. In *ACM SIGIR Forum*, volume 51, pages 260–267. ACM New York, NY, USA.

Jinhyuk Lee, Seongjun Yun, Hyunjae Kim, Miyoung Ko, and Jaewoo Kang. 2018. Ranking paragraphs for improving answer recall in open-domain question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 565–569. Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 3045–3059. Association for Computational Linguistics.

Shaobo Li, Xiaoguang Li, Lifeng Shang, Xin Jiang, Qun Liu, Chengjie Sun, Zhenzhou Ji, and Bingquan Liu. 2021. Hopretriever: Retrieve hops over wikipedia to answer complex questions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13279–13287.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*.

Yixin Nie, Songhe Wang, and Mohit Bansal. 2019. Revealing the importance of semantic retrieval for machine reading at scale. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 2553–2566. Association for Computational Linguistics.

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document ranking with a pre-trained sequence-to-sequence model. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 708–718. Association for Computational Linguistics.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*.

Ethan Perez, Patrick S. H. Lewis, Wen-tau Yih, Kyunghyun Cho, and Douwe Kiela. 2020. Unsupervised question decomposition for question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 8864–8880. Association for Computational Linguistics.

Jay M Ponte and W Bruce Croft. 2017. A language modeling approach to information retrieval. In *ACM SIGIR Forum*, volume 51, pages 202–208. ACM New York, NY, USA.

Peng Qi, Haejun Lee, Tg Sido, and Christopher D Manning. 2021. Answering open-domain questions of varying reasoning steps from text. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3599–3614.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Ori Ram, Gal Shachaf, Omer Levy, Jonathan Berant, and Amir Globerson. 2022. Learning to retrieve passages without supervision. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2687–2700, Seattle, United States. Association for Computational Linguistics.

Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at trec-3. *Nist Special Publication Sp*, 109:109.

Devendra Singh Sachan, Mike Lewis, Mandar Joshi, Armen Aghajanyan, Wen-tau Yih, Joelle Pineau, and Luke Zettlemoyer. 2022. Improving passage retrieval with zero-shot question generation. *arXiv preprint arXiv:2204.07496*.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

Timo Schick and Hinrich Schütze. 2021a. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 255–269. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021b. Few-shot text generation with natural language instructions. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 390–402. Association for Computational Linguistics.

Yeon Seonwoo, Sang-Woo Lee, Ji-Hoon Kim, Jung-Woo Ha, and Alice Oh. 2021. Weakly supervised pre-training for multi-hop retriever. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 694–704. Association for Computational Linguistics.

Devendra Singh, Siva Reddy, Will Hamilton, Chris Dyer, and Dani Yogatama. 2021. End-to-end training of multi-document reader and retriever for open-domain question answering. In *Advances in Neural Information Processing Systems*, volume 34, pages 25968–25981. Curran Associates, Inc.

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

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Wenhan Xiong, Xiang Lorraine Li, Srini Iyer, Jingfei Du, Patrick S. H. Lewis, William Yang Wang, Yashar Mehdad, Scott Yih, Sebastian Riedel, Douwe Kiela, and Barlas Oguz. 2021. Answering complex open-domain questions with multi-hop dense retrieval. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. 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, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Chen Zhao, Chenyan Xiong, Jordan L. Boyd-Graber, and Hal Daumé III. 2021a. Multi-step reasoning over unstructured text with beam dense retrieval. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 4635–4641. Association for Computational Linguistics.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021b. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 12697–12706. PMLR.

A Instructions

A.1 Best Instructions

Table A1 shows the top 10 performing instructions found by instruction search (§2.2) based on R@2 and using T5-XL.

A.2 Instruction Search

The actual templates we feed T5 are *"Task: <X> documents <Y>* question based on them. Question:" and *"Task: <X> previous documents and <Y> question based on them. Question:"*. We have found using the phrase "based on them" to be essential in directing the model to generate sensible instructions. Otherwise, the model would generate something like "Read the documents in question..". However, we remove that phrase from the obtained instructions".

B Document Scores

It is not immediately obvious how to compute a final score for each document since PROMPTRANK is mainly used to score document. The main issue is that a document can fall on multiple paths at the same time (some of which could be incomplete or not fully expanded yet) and therefore could have multiple such scores.

For example, assume a path $A \rightarrow B \rightarrow C$ of consisting of the documents A, B, and C, respectively. Considering the document B, we see that two scores are associated with B: score of the subpath $A \rightarrow B$ and score of the full $A \rightarrow B \rightarrow C$ path. To compute the final score of B, we could either just take the score of the longest path, or combine the two scores using mean, minimum, or maximum operations. What we found to work best compared to other alternatives is to take *maximum*, which is similar to what is done in (Das et al., 2019). We use this formulation when computing our recall metrics in §3.1.

C Hyperparameters

C.1 ELECTRA Reader

We use the same reader setting as in Xiong et al. (2021), where the top-100 retrieved paths are fed to the reader to obtain an answer from each path. Answers are then sorted based on a linear combination of path score and answer confidence, and the top answer is returned. We use the default hyperparameters for HotpotQA from (Xiong et al., 2021) in their codebase.[9] We use a maximum path length of 512 tokens, maximum question length of 64, and answer length of 30. In their experiments, Xiong et al. (2021) combine the answer confidence along with a ranking score using linear interpolation with a hyperparameter λ. For our experiments, we use the path scores produced by PROMPTRANK instead and learn λ on a held-out development set. The value we end up using for λ is 0.9.

D Further Results and Analysis

D.1 Results on 2WikiMQA

Table A2 shows the performance with few-shot PROMPTRANK with only one setting: few-shot, best instruction and with temperature scaling on the 2WikiMQA dataset (Ho et al., 2020). We compare it to two unsupervised baselines, namely TF-IDF and TF-IDF+BM25 . PROMPTRANK is significantly outperforming the baselines while using only 128 examples for tuning the instruction and the temperature parameter.

D.2 Inference Cost

Here, we analyze inference cost in terms of latency per query. We run retrieval using each method over 100 queries and then compute the average time per query. Inference was run over a single Nvidia Quadro RTX 8000 GPU. We run each method with the maximum batch size that fits within the GPU. One parameter that highly affects the speed for both PathRetriever and MDR is the beam size. We use the default beam size for PathRetriever, which is 8, and we use a beam size of 5 for MDR, to closely match PROMPTRANK's pruning parameter $K_1 = 5$. Other than beam size, we use the default parameters for each method.

Table A3 shows the number of parameters of each method and the average time per query in seconds. First, we note that PROMPTRANK uses the most number of parameters since it is based on T5-XLwhile PathRetriever and MDR both rely on much smaller LMs such as BERT and RoBERTa. Interestingly, however, we can see that PROMPTRANK without ensembling has lower latency than PathRetriever, which is slowed down by the beam search process since it has to expand and encode outgoing links from each passage in the beam at each step. As expected, ensembling almost multiplies the latency of PROMPTRANK by

[9] https://github.com/facebookresearch/multihop_dense_retrieval

ID	Prompt
1	Document: [D1] Document: [D2] , ..., Review previous documents and ask some question. Question:
2	Document: [D1] Document: [D2] , ..., Review the previous documents and answer question. Question:
3	Document: [D1] Document: [D2] , ..., Read the previous documents and write the following question. Question:
4	Document: [D1] Document: [D2] , ..., Search previous documents and ask the question. Question:
5	To analyze the documents and ask question. Document: [D1] Document: [D2] , ..., Question:
6	Document: [D1] Document: [D2] , ..., To read the previous documents and write a question. Question:
7	Document: [D1] Document: [D2] , ..., Read previous documents and write your exam question. Question:
8	Document: [D1] Document: [D2] , ..., Read the previous documents and ask this question. Question:
9	Read two documents and answer a question. Document: [D1] Document: [D2] , ..., Question:
10	Identify all documents and ask question. Document: [D1] Document: [D2] , ..., Question:

Table A1: Top 10 instructions found through automated instruction search (§2.2) using T5-XL. Instructions are sorted in descending order according to R@2 on a held-out development set of size 128 from HotpotQA (Yang et al., 2018). We use the first 5 for instruction ensembling (section §2.2). Blue represents fixed text that does not depend on the path *i.e* the instruction. The tokens [D1], [D2],.., etc. indicate where path documents are inserted.

	R@2	R@10	R@20	AR@2	AR@10	AR@20
Unsupervised Baselines						
TF-IDF	5.8	8.9	23.8	22.2	37.8	44.2
TF-IDF+BM25	7.6	34.1	45.8	20.7	44.9	53.0
PromptRank, *no ICL*						
T5-XL, best inst., temp. scaling	19.3	58.6	62.7	33.9	60.0	64.2

Table A2: Retrieval performance on 2WikiMQA (Ho et al., 2020).

the number of ensembles. Lastly, we note that MDR has significantly lower latency than both models, i.e., about 60x faster than PathRetriever and 46x than PromptRank, which is mainly due to the fast implementation of the exact inner product search (Johnson et al., 2019). It is worth noting, however, that MDR requires an expensive indexing step where *every* document in the corpus (Wikipedia in our case) is encoded using the document encoder. PromptRank, on the other hand, can work directly out-of-the-box without requiring such expensive indexing.

D.3 Sensitivity to Document Order

Here, study PromptRank's recall sensitivity to the document order in the prompt τ_c by running a simple experiment comparing two document ordering schemes: **link-based** and **inverted link-based**. Link-based ordering is the standard approach used in PromptRank, which orders the documents in

System	#params	Avg. query time(s)
PathRetriever	110M	1.95
MDR	125M	0.03
PromptRank	3B	1.38
PromptRank (ens)	3B	5.22

Table A3: Inference cost and number of parameters of three systems comparing PromptRank to PathRetriever and MDR. Query time is obtained by averaging the time to process 100 queries.

Doc. ordering	R@2	R@10	AR@2	AR@10
T5-Large				
Link-based	44.9	66.9	73.6	88.0
Inverted	44.5	67.7	72.6	87.8
T5-XL				
Link-based	44.6	67.9	74.1	88.2
Inverted	45.7	69.0	74.4	88.3

Table A4: Retrieval performance of PromptRank using two different orderings of the documents in the prompt. Evaluation is done on a set of 2K examples from HotpotQA train set. PromptRank exhibits minimal sensitivity to the document ordering.

the path based on their Wikipedia hyperlink traversal order. The inverted scheme, *reverses* the order of the documents in the prompt. No instruction is used for this experiment.

Table A4 shows the retrieval performance with both orderings. Interestingly, reversing the order of the documents in the path does not seem to have a tangible effect on the reranking performance. While it is expected that $p(q|\tau_c)$ will change by reversing the document order in the prompt, it appears that the ranks of different paths remain almost unchanged, which explains why the recall is hardly affected.

In other words, the path scores output by T5-XL does not appear to be sensitive to the document order prompt and can still. This might point to another benefit of LM-based path reranking: Since the performance is hardly affected by the document order, we do not have to worry about finding paths in the correct order (if such order exists) since the LM will still be able to assess the path relevance given different orders.

D.4 Comparison to Few-shot Systems

So far, we have mainly compared PromptRank to systems trained on many more examples. Here we compare PromptRank to few-shot LOUVRE (Seonwoo et al., 2021) and PathRetriever (Asai et al., 2020). To this end, we train PathRetriever on N examples from HotpotQA for $N \in 50, 100, 500, 1000$ and compare its performance to PromptRank ($N_{\text{demos}} = 10$). Since we were unable to obtain good performance by fine-tuning LOUVRE on few examples, we directly compare to the results reported in their paper, where 1% of training data is used (~90 examples). Table A5 shows performance of both few-shot systems compared to PromptRank. While PathRetriever's performance improves as we add more examples, we can see that it is much less data efficient than PromptRank. Even with 1K examples i.e., around 10x more data than PromptRank, it performs significantly worse across all metrics. We also observe that PromptRank performs better than LOUVRE in terms of R@2 and AR@2 (more than 6 points better) and very close with respect to other metrics even though PromptRank does *not* involve any (pre)training.

Approach	# Ex	R@2	R@10	AR@2	AR@10
PathRetriever	50	7.1 (4.4)	14.5 (4.6)	29.5 (6.2)	40.0 (3.7)
	100	10.8 (1.1)	19.1 (0.3)	34.8 (1.5)	43.1 (0.6)
	500	15.7 (0.3)	22.4 (0.3)	40.4 (0.3)	46.4 (0.4)
	1K	17.7 (0.4)	23.6 (0.3)	41.8 (0.3)	47.1 (0.3)
LOUVRE	1%	53.5	75.5	72.3	–
PromptRank ($N_{\text{demos}} = 10$)	128	54.4	73.5	78.9	88.9

Table A5: Retrieval performance of PromptRank compared to Few-shot PathRetriever. We show mean and (std) of PathRetriever's performance over 5 different seeds. The results of LOUVRE are take directly from (Seonwoo et al., 2021). We observe that PathRetriever performs very poorly in low-data settings, even when using about 10x more data than PromptRank. T

ACL 2023 Responsible NLP Checklist

A For every submission:

□ A1. Did you describe the limitations of your work?
Left blank.

□ A2. Did you discuss any potential risks of your work?
Left blank.

□ A3. Do the abstract and introduction summarize the paper's main claims?
Left blank.

□ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B □ Did you use or create scientific artifacts?

Left blank.

□ B1. Did you cite the creators of artifacts you used?
Left blank.

□ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Left blank.

□ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Left blank.

□ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Left blank.

□ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Left blank.

□ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Left blank.

C □ Did you run computational experiments?

Left blank.

□ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Left blank.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

□ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Left blank.

□ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Left blank.

□ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Left blank.

D □ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

□ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Left blank.

□ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Left blank.

□ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Left blank.

□ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Left blank.

□ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Left blank.