In-Context Retrieval-Augmented Language Models

Ori Ram[*] **Yoav Levine**[*] **Itay Dalmedigos** **Dor Muhlgay**
Amnon Shashua **Kevin Leyton-Brown** **Yoav Shoham**
AI21 Labs
`{orir,yoavl}@ai21.com`

Abstract

Retrieval-Augmented Language Modeling (RALM) methods, which show the LM relevant documents from a grounding corpus during generation, can mitigate the problem of factually inaccurate text generation. Existing RALM approaches focus on modifying the LM architecture in order to facilitate the incorporation of external information, significantly complicating deployment. This paper proposes an under-explored alternative, which we dub *in-context RALM*: leaving the LM architecture unchanged and prepending grounding documents to the input. We show that in-context RALM which uses off-the-shelf general purpose retrievers provides surprisingly large LM gains across five diverse corpora. We also demonstrate that the document retrieval and ranking mechanism can be specialized to the in-context RALM setting to further boost performance. We conclude that in-context RALM has considerable potential to increase the prevalence of LM grounding, particularly in settings where a pretrained LM must be used without modification or even via API access. To that end, we release all code, datasets, retrieval corpora, and indexes used in this paper.

1 Introduction

Recent advances in language modeling (LM) have dramatically increased the usefulness of machine-generated text across a wide range of use-cases and domains (Brown et al., 2020). One key challenge is that LM generated text often includes factual inaccuracies or errors (Maynez et al., 2020; Huang et al., 2020). This problem is present in any LM generation scenario, and is exacerbated when generation is made in uncommon domains, or when it involves up-to-date information that the LM has not seen during training. A promising approach for addressing this challenge is Retrieval-Augmented Language Modeling (RALM), grounding the LM during generation by conditioning on relevant documents retrieved from an external knowledge source. RALM systems include two high level components: (i) *document retrieval*, or selecting the set of documents upon which to condition; and (ii) *document reading*, or determining how to incorporate the selected documents into the LM generation process.

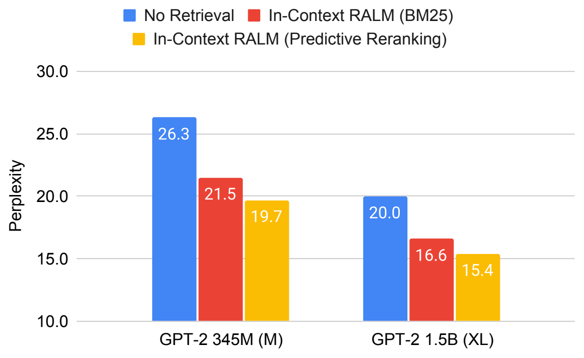

Figure 1: Our framework, dubbed *In-Context RALM*, provides large language modeling gains on the test set of WikiText-103, *without modifying the LM*. Adapting the use of a BM25 retriever (Robertson and Zaragoza, 2009) to the LM task (§5) yields significant gains, and choosing the grounding documents via our new class of Predictive Rerankers (§6) provides a further boost. See Table 1 for the full results on five diverse corpora.

Leading RALM systems introduced in recent years tend to be focused on altering the language model architecture (Khandelwal et al., 2020; Borgeaud et al., 2022; Zhong et al., 2022; Levine et al., 2022c; Li et al., 2022). Notably, Borgeaud et al. (2022) introduced RETRO, featuring document reading via nontrivial modifications to the LM architecture (and subsequent fine-tuning), while using an off-the-shelf frozen BERT retriever. Although the paper's experimental findings showed impressive performance gains, the need for changes in architecture and dedicated retraining has hindered the wide adoption of such models.

In this paper, we show that substantial gains can

[*]Equal contribution.

also be made by adapting the document selection mechanism to the task of language modeling, making it possible to achieve many of the benefits of RALM while working with off-the-shelf LMs, even via API access. Specifically, we propose a simple but powerful RALM framework, dubbed *in-context RALM* (presented in Section 3), which employs a zero-effort document integration mechanism: we simply prepend the selected documents to the LM's input text.

Section 4 describes our experimental setup. To show the wide applicability of our framework, we performed LM experiments on a suite of five diverse corpora: WikiText-103 (Merity et al., 2016), RealNews (Zellers et al., 2019), and three diverse datasets from The Pile (Gao et al., 2021), ArXiv, Stack Exchange and FreeLaw. We use open-source LMs with 110M–6B parameters (from the GPT-2 and GPT-Neo model families); we deliberately stay away from proprietary models such as Jurassic-1 (Lieber et al., 2021) to enable reproducibility.

In Section 5 we evaluate the application of off-the-shelf retrievers to the In-Context RALM setting, finding in this minimal effort setting that In-Context RALM led to LM performance gains equivalent to increasing the LM's number of parameters by 2–3$\times$ across all of the text corpora we examined. In Section 6 we investigate methods for adapting document ranking to the LM task, a relatively under-explored RALM degree of freedom. Our adaptation methods range from using a small LM to perform zero-shot ranking of the retrieved documents, up to training a dedicated bidirectional reranker by employing *self-supervision from the LM signal*. These methods lead to further gains in the LM task corresponding to an additional size increase of 2$\times$ in the LM architecture. As a concrete example of the gains, a 345M parameter GPT-2 enhanced by in-context RALM outperforms a 762M parameter GPT-2 when employing an off-the-shelf BM25 retriever (Robertson and Zaragoza, 2009), and outperforms a 1.5B parameter GPT-2 when employing our trained LM-oriented reranker.

We believe that in-context RALM can play two important roles in making RALM systems more powerful and more prevalent. First, given its simple reading mechanism, in-context RALM can serve as a clean probe for developing document retrieval methods that are specialized for the LM task. These in turn can be used to improve both in-context RALM and other more elaborate RALM methods that currently leverage general purpose retrievers. Second, due to its compatibility with off-the-shelf LMs, in-context RALM can help drive wider deployment of RALM systems. Despite the large gains shown by existing influential RALM papers, there is currently no RALM system that is widely used to augment off-the-shelf LMs. To help others both to deploy and to build upon our work, this paper is accompanied by an online release of all our code, datasets, trained models, and indexes for our standardized suite of corpora.

2 Related Work

The tendency of language models to produce factually-incorrect outputs has given rise to a line of work investigating RALM systems. Roughly speaking, RALM approaches can be divided into two families of models: (i) *nearest-neighbor language models* (also called kNN-LM), and (ii) *retrieve and read models*. Our work belongs to the second family, but is distinct in that it involves no further training of the LM.

Nearest Neighbor Language Models The kNN-LM approach was first introduced in Khandelwal et al. (2020). The authors suggest a simple inference-time model that interpolates between two next-token distributions: one induced by the LM itself, and one induced by the k neighbors from the retrieval corpus that are closest to the query token in the LM embedding space. Zhong et al. (2022) suggest a framework for training these models. While they showed significant gains from kNN-LM, the approach requires storing the representations *for each token in the corpus*, an expensive requirement even for a small corpus like Wikipedia. Although numerous approaches have been suggested for alleviating this issue (He et al., 2021; Alon et al., 2022), scaling any of them to large corpora remains an open challenge.

Retrieve and Read Models This family of RALMs creates a clear division between *document retrieval* and *document reading* components. All prior work involves training. Lewis et al. (2020) and Izacard and Grave (2021) fine tune a vanilla (*i.e.*, non-RALM) encoder–decoder into a RALM dedicated to downstream knowledge intensive tasks. Izacard et al. (2022b) explored different ways of pretraining such models, while Levine et al. (2022c) pretrained a autoregressive LM on clusters of nearest neighbors in sentence embedding space. Levine et al. (2022a,b) showed competitive open domain question-answering performance by prompt-tuning. Guu et al. (2020) pretrained REALM, a retrieval augmented *bidirectional, masked* LM, later fine-tuned for open-domain question answering. The work closest to this paper is RETRO (Borgeaud et al., 2022), which modifies an autoregressive LM (i.e., introduces new parameters to the model) to attend to relevant documents via chunked cross-attention. In contrast, our In-Context RALM approach applies *off-the-shelf* language models for document reading and does not require further training of the LM. In addition, we focus on how to choose documents for improved performance, an aspect not yet investigated by any of this prior work.

3 Our Framework: In-Context RALM

We now turn to describing our framework, *in-context retrieval augmented language modeling*. We begin by formulating RALM in general in Section 3.1, and then proceed to defining In-Context RALM in Section 3.2. At a high level, we emphasize the simplicity of the approach: In-Context RALM is simply about finding documents relevant to the LM generation, and prepending them as input to the LM, without further changing its operation.

3.1 Retrieval Augmented Language Modeling

Language models define probability distributions over sequences of tokens. Given such a sequence $x_1, ..., x_n$, the standard way to model its probability is via next-token prediction:

$$p(x_1, ..., x_n) = \prod_{i=1}^{n} p(x_i | x_{<i}), \qquad (1)$$

where $x_{<i} := x_1, ..., x_{i-1}$ is the sequence of tokens preceding x_i, also referred to as its *prefix*. This autoregressive model is usually implemented via a learned transformer network (Vaswani et al., 2017) parameterized by the set of parameters θ, which models the conditional probabilities $p_\theta(x_i | x_{<i})$ for $i \in [n] := \{1, ..., n\}$ by employing a causal self-attention mask (Radford et al., 2018).

Retrieval augmented language models (RALMs) add an operation that retrieves one or more documents from an external corpus $\mathcal{C}$, and condition the above LM predictions on these documents. Specifically, for predicting x_i, the retrieval operation from $\mathcal{C}$ depends on its prefix: $\mathcal{R}_\mathcal{C}(x_{<i})$, so the most general RALM decomposition is:

$$p(x_1, ..., x_n) = \prod_{i=1}^{n} p(x_i | x_{<i}, \mathcal{R}_\mathcal{C}(x_{<i})). \qquad (2)$$

We explicitize two practical relaxations often made in RALM systems in the above formulation.

Retrieval Stride While in the above formulation a retrieval operation can occur at each generation step, we might want to perform retrieval only once every $s > 1$ steps due to the cost of calling the retriever. We refer to s as the *retrieval stride*. This gives rise to the following RALM formulation (which reduces back to Eq. 2 for $s = 1$):

$$p(x_1, ..., x_n) = \prod_{i=0}^{n_s-1} \prod_{j=1}^{s} p\left(x_{s\cdot i+j} | x_{<(s\cdot i+j)}, \mathcal{R}_\mathcal{C}(x_{\leq s\cdot i})\right), \qquad (3)$$

where $n_s = n/s$ is the number of retrieval strides.

Retrieval Query Length While the retrieval query above in principle depends on all prefix tokens $x_{\leq s\cdot i}$, the information at the very end of the prefix is typically the most relevant to the generated tokens. If the retrieval query is too long then this information can be diluted. To avoid this, we restrict the retrieval query at stride i to the last ℓ tokens of the prefix, *i.e.*, we use $q_i^{s,\ell} := x_{s\cdot i-\ell+1}, ..., x_{s\cdot i}$. We refer to ℓ as the *retrieval query length*. Note that prior RALM work couples the retrieval stride s and the retrieval query length ℓ (Borgeaud et al., 2022). In Section 5, we study the effect of s and ℓ on the LM performance, and show that enforcing $s = \ell$ degrades performance. Integrating these hyper-parameters into the RALM formulation gives

$$p(x_1, ..., x_n) = \prod_{i=0}^{n_s-1} \prod_{j=1}^{s} p\left(x_{s\cdot i+j} | x_{<(s\cdot i+j)}, \mathcal{R}_\mathcal{C}(q_i^{s,\ell})\right). \qquad (4)$$

In order to condition the LM generation on the retrieved document, previous RALM approaches used specialized architectures or algorithms (§2), while In-Context RALM uses a regular unaltered LM.

3.2 In-Context Retrieval Augmented Language Modeling

We now define in-context RALM, perhaps the simplest approach for implementing the RALM formulation described above. Let $x := x_1, ..., x_{s\cdot i}$ be the prefix and $y := x_{s \cdot i+1}, ..., x_{s \cdot i+s}$ be the tokens in the upcoming stride. Also, let $d := z_1, ..., z_m$ be the document resulting from the retrieval operation, *i.e.*, $d = \mathcal{R}_{\mathcal{C}}(q_i^{s,\ell})$, where m is its length in tokens. To condition the LM on d, we simply concatenate d and x in the Transformer input, obtaining:

$$p\left(y|x, d\right) = \prod_{j=1}^{s} p_\theta(x_{s \cdot i+j}|z_1, ..., z_m, x_1, ..., x_{s \cdot i+j-1}) \tag{5}$$

where (crucially) p_θ is parameterized simply by an existing LM.

Modern, transformer-based LM architectures support limited-length inputs. Let w be the maximum input length supported by our LM. If $m+n > w$ (*i.e.*, if the concatenation of d and x would exceed the maximum input length), we remove tokens from the head of x until the overall input length equals that allowed by the model. In other words, in this case we take the last $w-m$ tokens. Since our documents are actually passages of limited length, in practice m is much smaller than w (in our experiments $m < 256$ and $w = 1024$; *cf.* Section 4.3), hence we always have enough context left from x.

4 Experimental Details

We now describe our experimental setup, including all models we use and their implementation details.

4.1 Datasets

We evaluated the effectiveness of in-context RALM across five diverse datasets. The first is **WikiText-103** (Merity et al., 2016), which has been extensively used to evaluate RALMs (Khandelwal et al., 2020; He et al., 2021; Borgeaud et al., 2022; Alon et al., 2022; Zhong et al., 2022). Second, we chose three datasets spanning diverse subjects from The Pile (Gao et al., 2021): **ArXiv**, **Stack Exchange** and **FreeLaw**. Finally, we also investigated **Real-News** (Zellers et al., 2019), since The Pile lacks a corpus focused only on news.

4.2 Models

Language Models We performed our experiments using variants of GPT-2 (Radford et al., 2019) and GPT-Neo (Black et al., 2021; Wang and Komatsuzaki, 2021): GPT-2 117M (also referred to as GPT-2 S or GPT-2 Small), GPT-2 345M (Medium), GPT-2 762M (Large), GPT-2 1.5B (XL),[1] GPT-Neo 1.3B, GPT-Neo 2.7B and GPT-J 6B. All models are open source and publicly available.[2] We elected to study these particular models for the following reasons. The first four (GPT-2) models were trained on WebText (Radford et al., 2019), with Wikipedia documents excluded from their training datasets. We were thus able to evaluate our method's "zero-shot" performance when retrieving from a novel dataset, WikiText-103. The three GPT-Neo models brought two further benefits. First, they allowed us to investigate how our methods scale to models larger than GPT-2. Second, the fact that Wikipedia was part of their training data allowed us to investigate the usefulness of in-context RALM for corpora seen during training. We observe that the helpfulness of such retrieval has been demonstrated for previous RALM methods (Khandelwal et al., 2020) and has also been justified theoretically by Levine et al. (2022c).

To facilitate more direct comparison between the models, we ran all of them with a maximum sequence length of 1,024 (*i.e.*, $w = 1024$), even though GPT-Neo models support a sequence length of 2,048.

Retrievers We experimented with both sparse (word-based) and dense (neural) retrievers. We used BM25 (Robertson and Zaragoza, 2009) as our sparse model. For dense models, we experimented with (i) a frozen BERT-base (Devlin et al., 2019) followed by mean pooling, similar to Borgeaud et al. (2022); and (ii) the Contriever (Izacard et al., 2022a) and Spider (Ram et al., 2022) models, which are unsupervised dense retrievers.

Reranking When training rerankers (Section 6.2), we initialized from RoBERTa-base (Liu et al., 2019), which shares the same vocabulary with our GPT-2 and GPT-Neo LMs.

4.3 Implementation Details

We implemented our code base using the Transformers library (Wolf et al., 2020). We based our

[1] Note that we were not able to reproduce the results that were reported for GPT-2 1.5B (XL) in Radford et al. (2019) on WikiText-103. The paper authors share that this inconsistency is due to their use of de-tokenizers for LM datasets: https://www.reddit.com/r/MachineLearning/comments/oye64h/comment/h7ucco2/?utm_source=share&utm_medium=web2x&context=3

[2] All models can be found and used via https://huggingface.co/

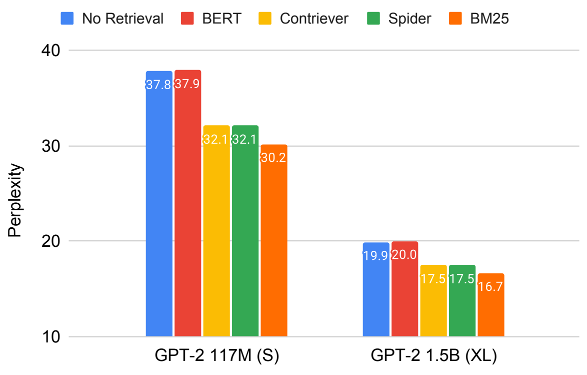

Figure 2: The performance of four off-the-shelf retrievers used for in-context RALM, on the development set of WikiText-103. The three RALMs are run with $s = 4$ (*i.e.*, retrieval is applied every four tokens). For each RALM, we report the result of the best query length ℓ.

dense retrieval code on the DPR repo (Karpukhin et al., 2020).

Retrieval Corpora For WikiText-103, we used the Wikipedia corpus from Dec. 20, 2018, standardized by Karpukhin et al. (2020) using the preprocessing from Chen et al. (2017). To avoid contamination, we found and removed all 120 articles of the development and test set of WikiText-103 from the corpus. For the remaining datasets, we used their training data as the retrieval corpus. Similar to Karpukhin et al. (2020), our retrieval corpora consist of non-overlapping passages of 100 words (which translate to less than 150 tokens for the vast majority of passages). Thus, we limit m (Section 3.2) to be 256, but it is usually much smaller.

Retrieval For sparse retrieval, we used the Pyserini library (Lin et al., 2021). For dense retrieval, we applied exact search using FAISS (Johnson et al., 2021).

5 The Effectiveness of In-Context RALM with Off-the-Shelf Retrievers

We now empirically show that despite its simple document reading mechanism, in-context RALM led to substantial LM gains across our diverse evaluation suite. We begin in this section by investigating the effectiveness of off-the-shelf retrievers for in-context RALM; we go on in Section 6 to show that further LM gains can be made by tailoring document ranking functions to the LM task.

The experiments in this section provided us with a recommended starting configuration for in-context RALM: applying a BM25 retriever that receives $\ell = 32$ token queries, with a retrieval frequency of every $s = 4$ tokens (ℓ and s are formally defined in Section 3). Tables 1 and 2 show that across all the examined corpora, employing in-context RALM with an off-the-shelf retriever improved LM perplexity to a sufficient extent that it matched that of a 2–3$\times$ larger model.

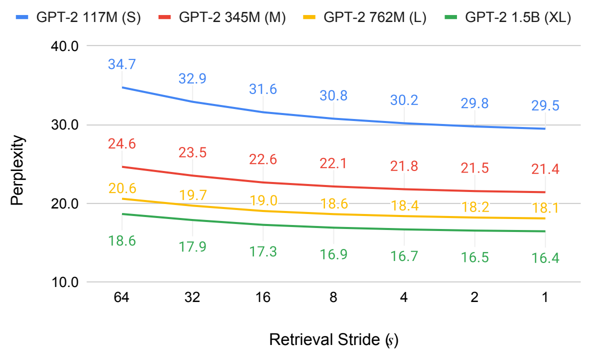

Figure 3: An analysis of perplexity as a function of *retrieval stride* (also referred to as s) on the development set of WikiText-103. Throughout the paper, we use $s = 4$ to balance perplexity and runtime.

5.1 BM25 Outperforms Off-the-Shelf Neural Retrievers in Language Modeling

We experimented with different off-the-shelf general purpose retrievers, and found that the sparse (lexical) BM25 retriever (Robertson and Zaragoza, 2009) outperformed three popular dense (neural) retrievers: a retriever based on the average pooling of BERT embeddings that was used in the RETRO paper (Borgeaud et al., 2022), and the strong self-supervised Contriever (Izacard et al., 2022a) and Spider (Ram et al., 2022).

Figure 2 compares the performance gains of in-context RALM with these four general-purpose retrievers. The BM25 retriever clearly outperformed all dense retrievers. This outcome is consistent with prior work showing that BM25 outperforms neural retrievers across a wide array of tasks (Thakur et al., 2021).

5.2 Frequent Retrieval Improves Language Modeling

We investigated the effect of varying the retrieval stride s (*i.e.*, the number of tokens between consecutive retrieval operations). Figure 3 shows that LM performance improved as the retrieval operation became more frequent. This supports the intuition that retrieved documents become more relevant the closer the retrieval query becomes to the generated

Model	Retrieval	Reranking	WikiText-103	RealNews	ArXiv	Stack Exch.	FreeLaw
			word ppl	token ppl	token ppl	token ppl	token ppl
GPT-2 110M (S)	–	–	37.5	21.3	11.6	16.9	13.6
	BM25 §5	–	29.6	16.1	10.5	14.6	9.2
	BM25	Zero-shot §6.1	28.6	15.5	9.6	13.7	8.4
	BM25	Predictive §6.2	26.8	–	–	–	–
GPT-2 345M (M)	–	–	26.3	15.7	9.0	11.0	9.8
	BM25 §5	–	21.5	12.4	8.3	10.0	7.0
	BM25	Zero-shot §6.1	20.8	12.0	7.6	9.6	6.5
	BM25	Predictive §6.2	19.7	–	–	–	–
GPT-2 762M (L)	–	–	22.0	13.6	8.2	10.4	9.1
	BM25 §5	–	18.1	10.9	7.5	9.5	6.7
	BM25	Zero-shot §6.1	17.6	10.6	7.0	9.1	6.2
	BM25	Predictive §6.2	16.6	–	–	–	–
GPT-2 1.5B (XL)	–	–	20.0*	12.4	7.6	10.0	8.5
	BM25 §5	–	16.6	10.1	7.0	9.2	6.3
	BM25	Zero-shot §6.1	16.1	9.8	6.5	8.8	5.9
	BM25	Predictive §6.2	15.4	–	–	–	–

Table 1: Perplexity on the test set of WikiText-103 and the development sets of RealNews and three datasets from the Pile, with and without using the top-scored passage retrieved by BM25. All models share the same vocabulary, thus token-level perplexity (*token ppl*) numbers are comparable. For WikiText we follow prior work and report word-level perplexity (*word ppl*). *Result is not consistent with Radford et al. (2019), please see Footnote 1.

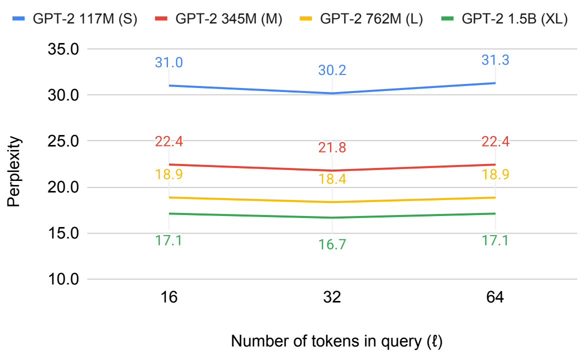

Figure 4: An analysis of perplexity as a function of *the number of tokens in the query* for BM25 on the development set of WikiText-103. In the appendix, we show similar patterns for other corpora in the "diverse corpora" suite, and a similar tradeoff for the dense contriever retriever within WikiText-103. Throughout the paper, we use 32 tokens in the query.

Model	Retrieval	WikiText-103	RealNews
		word ppl	token ppl
GPT-Neo 1.3B	-	17.5	12.3
	BM25, §5	14.6	9.9
GPT-Neo 2.7B	-	15.1	11.0
	BM25, §5	12.8	9.0
GPT-J 6B	-	11.6	9.2
	BM25, §5	10.0	7.7

Table 2: The performance of larger models from the GPT-Neo family, measures by word-level perplexity on the test set of WikiText-103 and token-level perplexity on the development set of RealNews.

tokens. Of course, each retrieval operation imposes a runtime cost. To balance performance and runtime, we used $s = 4$ in our experiments. For comparison, RETRO employed a retrieval frequency of $s = 64$ (Borgeaud et al., 2022).

5.3 A Contextualization vs. Recency Tradeoff in Query Length

We also investigated the effect of varying ℓ, the length of the retrieval query for BM25. Figure 4 reveals an interesting tradeoff and a sweet spot around a query length of 32 tokens. Similar experiments for dense retrievers are given in App. A. We conjecture that when the retriever query is too short, it does not include enough of the input context, decreasing the retrieved document's relevance. Conversely, excessively growing the retriever query deemphasizes the tokens at the very end of the prefix, diluting the query's relevance to the LM task.

6 Improving In-Context RALM with LM-Oriented Reranking

Since in-context RALM uses a fixed document reading component by definition, it is natural to ask whether its performance can be improved by adapting its document retrieval mechanism. Indeed, there is considerable scope for improvement: the

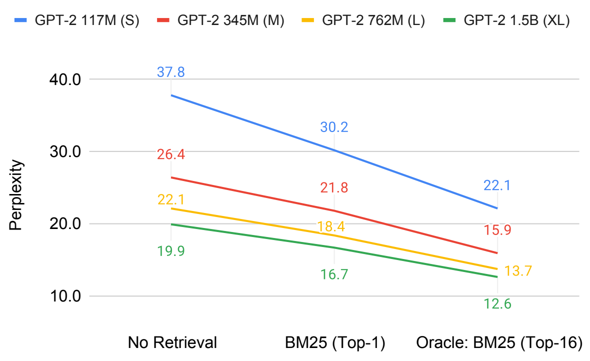

Figure 5: Potential for gains from reranking: perplexity improvement from an oracle that takes the best of the 16 top documents retrieved by BM25 rather than the first.

previous section considered conditioning the model only on the first document retrieved by the BM25 retriever. This permits very limited semantic understanding of the query, since BM25 is based only on the bag of words signal. Moreover, it offers no way to accord different degrees of importance to different query tokens, such as recognizing that later query tokens are more relevant to the generated text.

In this section, we focus on choosing which document to present to the model, reranking the top k documents returned by the BM25 retriever. Figure 5 shows the large potential for improvement among the top 16 documents returned by the BM25 retriever. Specifically, in Section 6.1, we show performance gains across our evaluation suite obtained by using an LM to perform zero-shot reranking of the top-k BM25 retrieved documents. Then, in Section 6.2, we show that training a specialized bidirectional reranker of the top-k BM25 retrieved documents over the LM signal can provide further LM gains.

6.1 LMs as Zero-Shot Rerankers

We used language models as document rerankers for the RALM. Formally, for a query q consisting of the last ℓ tokens in the prefix of the LM input x, let $\{d_1, ..., d_k\}$ be the top k documents returned by BM25. For retrieval iteration i, the text for generation is $y := x_{si+1}, ..., x_{si+s}$. Ideally, we would like to find the document d_{j^*} that maximizes the probability of the text for generation, *i.e.*,

$$j^* = \arg\max_{j \in [k]} p(y | x_{\leq si}, d_j) \tag{6}$$

according to Eq. (5). However, at test time, we do not have access to the tokens of y. Thus, we used the last *prefix* tokens instead. Formally, we define a hyper-parameter s' that determines the number of the prefix tokens by which to rerank, choosing document $d_{\hat{j}}$ such that

$$\hat{j} = \arg\max_{j \in [k]} p(x_{si-s'+1}, ..., x_{si} | x_{\leq (si-s')}, d_j). \tag{7}$$

The main motivation is that since BM25 is a lexical retriever, we want to incorporate a semantic signal induced by the LM. Also, this reranking shares conceptual similarities with the reranking framework of Sachan et al. (2022) for open-domain question answering, where our prefix $x_{\leq si}$ can be thought of as their "question".

Note that our zero-shot reranking does not require that the LM used for reranking (*i.e.*, in Eq. (7)) be the same as the actual language model (*i.e.*, in Eq. (5)). This observation unlocks the possibility of reranking with smaller (and thus faster) models, which is important for two main reasons: (i) Reranking k documents requires k forward passes; and (ii) it allows our methods to be used in cases where the actual LM's log probabilities are not available (for example, when the LM is accessed through an API).

Results A minimal hyper-parameter search on the development set of WikiText-103 revealed that the optimal query length is $s' = 16$,[3] so we proceed with this value going forward. Table 1 shows the results of letting the LM perform zero-shot reranking on the top-16 documents retrieved by BM25 (third row for each of the models). Table 3 in the appendix shows that the a small LM (GPT-2 117M) can be used to re-rank the documents for larger LMs (GPT-2 345M-1.5B), with roughly the same performance as having each LM perform reranking for itself, suppporting the applicability of this method for LMs that are only accessible via an API. Indeed, it is evident that reranking yielded consistently better results than simply taking the first result returned by the retriever. For example, the perplexity of GPT-2 117M on WikiText-103 improved from 29.6 to 28.6, and between 0.6 and 1.0 in the rest of the datasets. Overall, we observed improvements for all models and datasets.

6.2 Training LM-dedicated Rerankers

Next, we trained a reranker to choose the documents from the top-k documents retrieved by

[3]We experimented with $s' \in \{4, 8, 16, 32\}$.

BM25. We refer to this as a *predictive reranker*, since it learned to choose which document will help in predicting the upcoming text. For this process, we assume availability of training data from the target corpus. Our reranker is a classifier that gets a document d_j (for $j \in [k]$) and a prefix $x_{\leq s \cdot i}$, and produces a scalar $f(x_{\leq s \cdot i}, d_j)$ that should resemble the relevance of d_j for the continuation of $x_{\leq s \cdot i}$.

We then normalize these relevance scores:

$$p_{\text{rank}}(d_j | x_{\leq s \cdot i}) = \frac{\exp(f(x_{\leq s \cdot i}, d_j))}{\sum_{j'=1}^{k} \exp(f(x_{\leq s \cdot i}, d_{j'}))}, \quad (8)$$

and choose the document $d_{\hat{j}}$ such that

$$\hat{j} = \arg\max_{j \in [k]} \ p_{\text{rank}}(d_j | x_{\leq s \cdot i}). \quad (9)$$

Training Process Our reranker was a fine-tuned RoBERTa-base (Liu et al., 2019) trained as follows. Let $x_{\leq si}$ be a prefix we sample from the training data, and $y := x_{si+1}, ..., x_{si+s}$ its next stride. We run BM25 on the query q derived from $x_{\leq si}$ and get k documents $\{d_1, ..., d_k\}$. For each document d_j, we then run the LM to compute $p(y | x_{\leq s \cdot i}, d_j)$ according to Eq. 5. The objective function we use to train the reranker follows previous work (Guu et al., 2020; Lewis et al., 2020):

$$-\log \sum_{j=1}^{k} p_{\text{rank}}(d_j | x_{\leq s \cdot i}) \cdot p(y | x_{\leq s \cdot i}, d_j). \quad (10)$$

Note that unlike these two works, we train only the reranker (p_{rank}), while the LM is kept frozen. The motivation for using this objective, rather than a classification-like loss function like cross entropy, is that it does not enforce gradients when all k documents have similar $p(y | x_{\leq s \cdot i}, d_j)$ values (which is the case, for example, when none of them helps the model).

Results Table 1 shows the result of training a predictive reranker on the training set of WikiText-103. Specifically, we train it with data produced by GPT-2 110M (S), and test its effectiveness for all GPT-2 models. We observed significant gains obtained from predictive reranking. For example, the perplexity of GPT-2 110M (S) improved from 29.6 to 26.8, and that of GPT-2 1.5B (XL) improved from 16.6 to 15.4. This trend held for the other two models as well. Overall, these results demonstrate that training a reranker with domain-specific data was more effective than zero-shot reranking (Section 6.1). Note that these results—while impressive—still leave room for further improvements, compared to the oracle results (*cf.* Figure 5).

7 Discussion

Retrieval from external sources has become a common practice in knowledge-intensive tasks (such as factual question answering, fact checking, and more; Petroni et al. 2021). In parallel, recent breakthroughs in LM generation capabilities has led to LMs that can generate useful long texts. However, factual inaccuracies remain a common way in which machine-generated text can fall short. This makes RALMs both a promising and an urgent new application area for knowledge grounding. Prior research has already investigated this application, of course, but it is not yet widely deployed. One likely reason is that existing approaches rely upon fine-tuning the LM, which is typically difficult and costly, and is impossible for LMs accessible only via an API.

This paper presented the framework of *in-context RALM*, enabling frozen LMs to benefit from retrieval. We demonstrated that substantial performance gains can be achieved by integrating external knowledge into a frozen LM and showed that additional gains can be achieved from adjusting the retriever query to the LM task; tailoring the retrieved documents to the generation setting; and adapting the document reading mechanism to facilitate retrieval during generation.

Several directions for further improvement remain for future work. First, this paper considers only the case of prepending a single external document to the context; adding more documents could drive further gains. Second, we retrieved documents every fixed interval of s tokens during generation, but see potential for large latency and cost gains by retrieving more sparsely, such as only when a specialized model predicts that retrieval is needed. Finally, Ratner et al. (2022) recently propose a method of parallelizing the input sequence when generating with off-the-shelf LMs. This can potentially be applied in order to show retrieved documents in parallel to the prefix, rather than before it, possibly improving the utilization of the external knowledge during text generation. We release all resources used for this paper, for the community to use and improve over. We hope that these resources will drive further research of

RALMs, that will enable wider adoption.

References

Uri Alon, Frank Xu, Junxian He, Sudipta Sengupta, Dan Roth, and Graham Neubig. 2022. Neuro-symbolic language modeling with automaton-augmented retrieval. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 468–485. PMLR.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack Rae, Erich Elsen, and Laurent Sifre. 2022. Improving language models by retrieving from trillions of tokens. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240. PMLR.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2021. The pile: An 800gb dataset of diverse text for language modeling.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: Retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. 2021. Efficient nearest neighbor language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5703–5714, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Minlie Huang, Xiaoyan Zhu, and Jianfeng Gao. 2020. Challenges in building intelligent open-domain dialog systems. *ACM Trans. Inf. Syst.*, 38(3).

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022a. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, Online. Association for Computational Linguistics.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022b. Atlas: Few-shot learning with retrieval augmented language models.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick 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)*, pages 6769–6781, Online. Association for Computational Linguistics.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*.

Yoav Levine, Itay Dalmedigos, Ori Ram, Yoel Zeldes, Daniel Jannai, Dor Muhlgay, Yoni Osin, Opher Lieber, Barak Lenz, Shai Shalev-Shwartz, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham.

2022a. Standing on the shoulders of giant frozen language models.

Yoav Levine, Ori Ram, Daniel Jannai, Barak Lenz, Shai Shalev-Shwartz, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2022b. Huge frozen language models as readers for open-domain question answering. In *ICML 2022 Workshop on Knowledge Retrieval and Language Models*.

Yoav Levine, Noam Wies, Daniel Jannai, Dan Navon, Yedid Hoshen, and Amnon Shashua. 2022c. The inductive bias of in-context learning: Rethinking pretraining example design. In *International Conference on Learning Representations*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Zonglin Li, Ruiqi Guo, and Sanjiv Kumar. 2022. Decoupled context processing for context augmented language modeling. In *Advances in Neural Information Processing Systems*.

Opher Lieber, Or Sharir, Barak Lenz, and Yoav Shoham. 2021. Jurassic-1: Technical details and evaluation.

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 2356–2362, New York, NY, USA. Association for Computing Machinery.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized bert pretraining approach.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, Online. Association for Computational Linguistics.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. KILT: a benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2523–2544, Online. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

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.

Nir Ratner, Yoav Levine, Yonatan Belinkov, Ori Ram, Omri Abend, Ehud Karpas, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2022. Parallel context windows improve in-context learning of large language models.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

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. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. Beir: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 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, Online. Association for Computational Linguistics.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Zexuan Zhong, Tao Lei, and Danqi Chen. 2022. Training language models with memory augmentation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

A Query Length Ablations

Figure 6, Figure 7 and Figure 8 show ablations on the optimal query length ℓ for off-the-shelf dense retrievers (BERT, Contriever and Spider, respectively). Consistently, using $\ell = 64$ (tokens) is optimal. This is in contrast to similar experiments we conducted for BM25 (*cf.* Figure 4), where $\ell = 32$ is optimal.

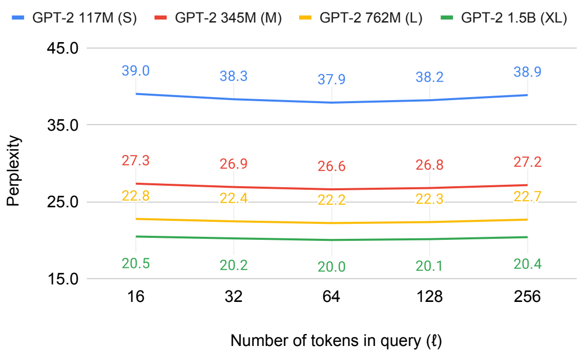

Figure 6: An analysis of perplexity as a function of *the number of tokens in the query* for an off-the-shelf BERT retriever on the development set of WikiText-103.

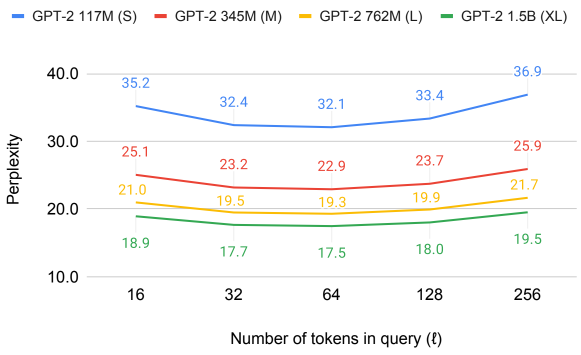

Figure 7: An analysis of perplexity as a function of *the number of tokens in the query* for Contriever on the development set of WikiText-103.

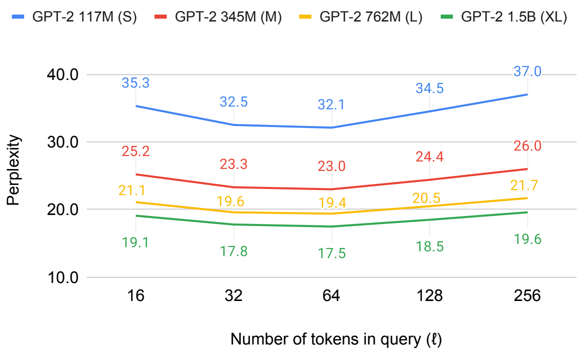

Figure 8: An analysis of perplexity as a function of *the number of tokens in the query* for Spider on the development set of WikiText-103.

Model	Reranking Model	WikiText-103	RealNews	Arxiv	Stack Exch.	FreeLaw
		word ppl	token ppl	token ppl	token ppl	token ppl
GPT-2 345M (M)	GPT-2 110M (S)	20.8	12.1	7.7	9.6	6.5
	GPT-2 345M (M)	20.8	12.0	7.6	9.6	6.5
GPT-2 762M (L)	GPT-2 110M (S)	17.7	10.7	7.0	9.1	6.3
	GPT-2 762M (L)	17.6	10.6	7.0	9.1	6.2
GPT-2 1.5B (XL)	GPT-2 110M (S)	16.2	9.9	6.6	8.9	6.0
	GPT-2 1.5B (XL)	16.1	9.8	6.5	8.8	5.9

Table 3: Perplexity for zero-shot reranking (§6.1) where the reranking models is smaller than the LM, or the LM itself. Reranking is performed on the top 16 documents retrieved by BM25. Using a GPT-2 110M (S) instead of a larger language model as a reranker leads to only a minor degradation.