Fine-Tuning LLaMA for Multi-Stage Text Retrieval

Xueguang Ma*
University of Waterloo
Waterloo, Canada
x93ma@uwaterloo.ca

Liang Wang
Microsoft Research Asia
Beijing, China
wangliang@microsoft.com

Nan Yang
Microsoft Research Asia
Beijing, China
nanya@microsoft.com

Furu Wei
Microsoft Research Asia
Beijing, China
fuwei@microsoft.com

Jimmy Lin
University of Waterloo
Waterloo, Canada
jimmylin@uwaterloo.ca

ABSTRACT

While large language models (LLMs) have shown impressive NLP capabilities, existing IR applications mainly focus on prompting LLMs to generate query expansions or generating permutations for listwise reranking. In this study, we leverage LLMs directly to serve as components in the widely used multi-stage text ranking pipeline. Specifically, we fine-tune the open-source LLaMA-2 model as a dense retriever (repLLaMA) and a pointwise reranker (rankLLaMA). This is performed for both passage and document retrieval tasks using the MS MARCO training data. Our study shows that fine-tuned LLM retrieval models outperform smaller models. They are more effective and exhibit greater generalizability, requiring only a straightforward training strategy. Moreover, our pipeline allows for the fine-tuning of LLMs at each stage of a multi-stage retrieval pipeline. This demonstrates the strong potential for optimizing LLMs to enhance a variety of retrieval tasks. Furthermore, as LLMs are naturally pre-trained with longer contexts, they can directly represent longer documents. This eliminates the need for heuristic segmenting and pooling strategies to rank long documents. On the MS MARCO and BEIR datasets, our repLLaMA–rankLLaMA pipeline demonstrates a high level of effectiveness.

CCS CONCEPTS

• **Information systems → Retrieval models and ranking**.

KEYWORDS

Large Language Model, Dense Retrieval, Reranker

ACM Reference Format:
Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2024. Fine-Tuning LLaMA for Multi-Stage Text Retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24), July 14–18, 2024, Washington, DC, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3626772.3657951

*Work done during Xueguang's internship at MSRA.

SIGIR '24, July 14–18, 2024, Washington, DC, USA

ACM ISBN 979-8-4007-0431-4/24/07
https://doi.org/10.1145/3626772.3657951

1 INTRODUCTION

Text retrieval is crucial in various natural language comprehension tasks [25], including web search [1], open-domain question answering [2], and fact verification [34]. Retrieval also plays an important role in enhancing the effectiveness of large language models (LLMs) in a retrieval-augmented generation (RAG) pipeline [15, 31]. This approach not only mitigates hallucinations but also enables LLMs to access external knowledge [12, 42].

A typical multi-stage text retrieval pipeline consists of a *retriever*, designed to efficiently locate the top-k relevant texts from a (potentially large) corpus, and a *reranker*, which further refines the order of the retrieved candidates to improve output quality [22]. Both retrievers and rerankers have significantly benefited from the advent of pre-trained language models based on Transformers [37] such as BERT [7] and T5 [30]. These models are fine-tuned to encode queries and documents into vector representations for retrieval [13, 16] or to directly score the relevance between a query and a document for reranking [23]. Several solutions have been introduced to enhance the effectiveness of retrievers and rerankers with improved data creation or training strategies [9, 29, 38, 40, 41, 44].

Recent LLMs with billions of parameters such as GPT-4 [24] and LLaMA [35, 36] have exhibited extraordinary capabilities in many NLP tasks, surpassing previous smaller models [43]. For retrieval, recent methods such as RankGPT [32], LRL [18], and PRP [28] have explored prompting LLMs to perform zero-shot listwise or pairwise ranking as text generation tasks. Work such as HyDE [10] and Query2Doc [39] have used LLMs to augment user queries. These methods rely on the strong generative capabilities of LLMs. However, we see a few potential issues. First, these methods do not address the entire multi-stage pipeline, as it is challenging to cast first-stage retrieval as text generation. Second, they do not leverage labeled data when available. Finally, prompting-based rerankers are not efficient because they do not support parallel scoring; also, their multi-pass decoding design and sliding window strategy [18, 32] present efficiency bottlenecks.

Therefore, we argue that fine-tuning state-of-the-art LLMs to function as retrievers and rerankers in multi-stage pipelines can yield better effectiveness than older, smaller models. Previous work such as GTR [21], SGPT [19], and cpt-text [20] discussed fine-tuning language models with billions of parameters to generate dense embeddings. However, LLaMA has demonstrated even better effectiveness on natural language generation tasks than previous open-source models, suggesting that it might serve as a better backbone and warranting further exploration. Additionally, none of the models referenced above are fully optimized for a multi-stage retrieval pipeline. Thus, we investigate the following research question: How do state-of-the-art LLMs perform when specifically fine-tuned for multi-stage text retrieval?

Our study answers this question by conducting a comprehensive investigation into fine-tuning LLaMA-2 [35] as both a dense retriever and a pointwise reranker, which we refer to as repLLaMA and rankLLaMA, respectively. We find that LLMs surpass previous smaller models in terms of effectiveness for both retrieval and reranking using only a straightforward training regime and exhibiting strong zero-shot effectiveness. Compared to methods that directly prompt large language models to generate permutations [32], fine-tuning large language models as rerankers can be more effective due to the use of labeled data and be more efficient due to decoupled parallel inference. Furthermore, we observe that LLMs, which are inherently pre-trained on longer contexts, are capable of representing entire documents, thereby eliminating the need for segmenting and pooling strategies for document retrieval.

2 METHOD

2.1 Retriever

Our retriever model, called repLLaMA, follows the bi-encoder architecture proposed in DPR [13], but with the backbone model initialized with LLaMA. In contrast to previous dense retriever models based on BERT, which take the representation of the prepended [CLS] token as the representation of the text input, repLLaMA computes the vector embedding of a query or a document as:

$$V_T = \text{Decoder}(\text{`}t_1\textvisiblespace t_2\textvisiblespace\ldots\textvisiblespace t_k\text{'})[-1]$$

where Decoder(·) represents the LLaMA model, which returns the last layer token representation for each input token. As the attention mechanism of LLaMA is uni-directional, we take the representation of the last token as the representation of the input sequence $t_1 \ldots t_k$, either a query Q or a document D.

Relevance of D to Q is computed in terms of the dot product of their corresponding dense representations V_Q and V_D as $\text{Sim}(Q, D) =< V_Q, V_D >$. The model is then optimized end-to-end according to InfoNCE loss with a set of negative documents that includes both hard negatives, which are sampled from the top-ranking results of an existing retrieval system, and in-batch negatives, which are derived from the positive documents and hard negative documents associated with other queries in the same training batch. In practice, dense retrieval training tends to benefit from a larger set of hard negatives and in-batch negatives.

2.2 Reranker

Our rankLLaMA reranker model is trained as a pointwise reranker. This approach involves passing a query and a candidate document together as model input, with the model generating a score that indicates the relevance of the document to the query [23]. In more detail, rankLLaMA reranks a query–document pair as follows:

$$\text{input} = \text{`query: }\{Q\}\text{ document: }\{D\}\text{'}$$
$$\text{Sim}(Q, D) = \text{Linear}(\text{Decoder}(\text{input})[-1])$$

where Linear(·) is a linear projection layer that projects the last layer representation of the last token to a scalar. Similar to the retriever, the model is optimized by contrastive loss, but, in this case, the negative documents do not involve in-batch negatives.

To train a reranker that is optimized to rerank candidates from a specific retriever in a multi-stage pipeline, hard negatives should be sampled from the top-ranking results from that retriever. Specifically, in our case, the hard negative training data for rankLLaMA are selected from the top-ranking results of repLLaMA.

3 EXPERIMENTS

3.1 Passage Retrieval

3.1.1 Dataset. We train our retriever and reranker models with LLaMA-2 on the training split of MS MARCO passage [1]. As discussed in Section 2.1, the use of hard negatives is crucial for effective training. We use a blend of BM25 and CoCondenser [9] hard negatives to ensure that the hard negatives are derived from both sparse and dense retrieval results, thereby enhancing the diversity of the samples. For the reranker, we select the hard negatives from the top-200 candidates generated by the retriever.

We evaluate the effectiveness of our models using the development split of the MS MARCO passage ranking task and the TREC DL19/DL20 passage ranking tasks [3, 4]. Following standard practice, we adopt MRR@10 and nDCG@10 as the evaluation metrics. In addition, we assess the zero-shot effectiveness of repLLaMA and rankLLaMA on the 13 publicly available datasets of BEIR [33].

3.1.2 Implementation Details. We initialize our models with the LLaMA-2-7B checkpoint[1] and train on 16 × 32G V100 GPUs. For repLLaMA, we append an end-of-sequence token </s> to the input sequence and take its final layer representation as the dense representation (4096 dimensions). Additionally, we normalize these dense representations into unit vectors during both the training and inference stages, ensuring that their L2-norms are equal to 1. After encoding the entire corpus, we end up with a 135G flat index for brute-force search.

For rankLLaMA, we find that appending </s> to the input sequence causes loss overflow error when fine-tuning LLaMA-2 with 16-bit floating point precision. Thus, we use the final layer representation of the last token in the passage to calculate the similarity score. A challenge in fine-tuning LLMs for retrieval is the high GPU memory costs associated with contrastive learning, as it requires large batch sizes for in-batch negatives. To address this, we employ recently proposed solutions, including LoRA [11], flash attention [6], and gradient checkpointing to reduce GPU memory usage. Both the retriever and reranker are trained with a batch size of 128, with 15 hard negative passages sampled for each query. At inference time, repLLaMA retrieves the top-1000 passages from the corpus and rankLLaMA reranks the top-200 passages retrieved by repLLaMA. To explore whether increases in model size can further improve effectiveness, we also train a version of rankLLaMA using LLaMA-2-13B as the model initialization.[2]

3.1.3 In-Domain Evaluation. As shown in Table 1, repLLaMA outperforms all competing methods for retrieval, achieving the highest

[1]https://huggingface.co/meta-llama/Llama-2-7b-hf
[2]https://huggingface.co/meta-llama/Llama-2-13b-hf

	Model size	Source prev.	top-k	Dev MRR@10	DL19 nDCG@10	DL20 nDCG@10
Retrieval						
(a) BM25 [17]	-	-	$\|C\|$	18.4	50.6	48.0
(b) ANCE [41]	125M	-	$\|C\|$	33.0	64.5	64.6
(c) CoCondenser [9]	110M	-	$\|C\|$	38.2	71.7	68.4
(d) GTR-base [21]	110M	-	$\|C\|$	36.6	-	-
(e) GTR-XXL [21]	4.8B	-	$\|C\|$	38.8	-	-
(f) OpenAI Ada2 [20]	?	-	$\|C\|$	34.4	70.4	67.6
(g) bi-SimLM [38]	110M	-	$\|C\|$	39.1	69.8	69.2
(h) repLLaMA	7B	-	$\|C\|$	**41.2**	**74.3**	**72.1**
Reranking						
(i) monoBERT [23]	110M	(a)	1000	37.2	72.3	72.2
(j) cross-SimLM [38]	110M	(g)	200	43.7	74.6	72.7
(k) RankT5 [44]	220M	(d)	1000	43.4	-	-
(l) rankLLaMA	7B	(h)	200	44.9	75.6	77.4
(m) rankLLaMA-13B	13B	(h)	200	**45.2**	**76.0**	**77.9**
(n) RankVicuna [27]	7B	(a)	100	-	66.8	65.5
(o) PRP [28]	20B	(a)	100	-	72.7	70.5
(p) RankGPT$_{3.5}$ [32]	?	(a)	100	-	65.8	72.9
(q) RankGPT$_4$ [32]	?	(p)	30	-	75.6	70.6

Table 1: The effectiveness of repLLaMA and rankLLaMA on the MS MARCO passage corpus compared to baselines.

Method	Model Size	BEIR-13 Avg.
BM25 [17]	-	43.7
GTR-XXL [21]	4.8B	49.3
Ada2 [20]	?	52.1
SGPT [19]	5.8B	52.1
repLLaMA	7B	**55.1**
RankT5 [44]	220M	53.7
rankLLaMA	7B	**56.6**
rankLLaMA-13B	13B	56.5

Table 2: Zero-shot effectiveness of repLLaMA and rankLLaMA on BEIR datasets.

effectiveness. Compared to GTR-XXL, which also uses a model with billions of parameters based on the T5-encoder [21], our model achieves 2.4 points higher MRR@10 on the dev queries.

For reranking, rankLLaMA reranks the top-200 passages from repLLaMA, resulting in the highest end-to-end effectiveness of any multi-stage retrieval system that we are aware of. Our complete repLLaMA–rankLLaMA pipeline beats (to our knowledge) the previous state-of-the-art reranker, RankT5 [44]. Furthermore, our rankLLaMA-13B model outperforms the 7B model, achieving slightly higher effectiveness. Compared to RankGPT$_4$ [32], which prompts GPT-4 to perform passage reranking through permutation generation within a multi-stage retrieval pipeline, our repLLaMA–rankLLaMA combination achieves higher nDCG@10 scores on both DL19 and DL20. As a pointwise reranker, rankLLaMA can rerank candidate passages in parallel, which means that inference can be accelerated to reduce latency as compared to RankGPT, which depends on a sequential sliding-window strategy to rerank.

3.1.4 Zero-Shot Evaluation. Results on the BEIR datasets are presented in Table 2. Both our models demonstrate superior zero-shot effectiveness, outperforming existing models. repLLaMA surpasses other existing dense retrievers with billions of parameters. Specifically, it outperforms SGPT and Ada2 as well as GTR-XXL. Note that Ada2 and GTR-XXL require an unsupervised contrastive pre-training stage before the supervised fine-tuning. In contrast, repLLaMA uses the base pre-trained model as initialization, achieving the highest zero-shot effectiveness we are aware of while maintaining simplicity. rankLLaMA-7B further enhances the retriever's effectiveness when reranking the top-100 retrieval results from repLLaMA, but interestingly, the larger rankLLaMA-13B model does not appear to yield any further improvements.

	Source prev.	top-k	Seg. Y/N	Dev MRR@100	DL19 nDCG@10	DL20 nDCG@10
Retrieval						
(a) BM25 [17]	-	$\|C\|$	N	23.0	51.8	52.9
(b) BM25-Q2D [26]	-	$\|C\|$	Y	31.8	61.2	59.6
(c) CoCondenser [9]	-	$\|C\|$	Y	42.5	64.8	**64.0**
(d) repLLaMA	-	$\|C\|$	N	**45.6**	**65.0**	63.2
Reranking						
(e) monoT5 [26]	(b)	10000	Y	41.1	-	-
(f) MORES+ [8]	(c)	100	Y	49.3	-	-
(g) rankLLaMA	(d)	100	N	**50.3**	**67.7**	**67.4**

Table 3: The effectiveness of repLLaMA and rankLLaMA on the MS MARCO document corpus compared to baselines.

3.2 Document Retrieval

3.2.1 Dataset. Document retrieval presents the challenge of handling long input sequences [1]; for example, the MS MARCO document ranking corpus has an average document length of around 1500 tokens. Notably, only 24% of the documents have fewer than 512 tokens, which is the maximum input length for most previous rankers based on smaller pre-trained language models [7].

The standard solution to manage long sequences for retrieval is the MaxP strategy [5], which involves dividing the document into segments and determining the document score based on the segment with the highest score. However, this involves a heuristic pooling strategy and runs the risk of losing information distributed across long contexts. Recent language models pre-trained on longer sequences (e.g., 4096 tokens for LLaMA-2) enable us to represent longer texts "in one go", eliminating the need for segmentation.

3.2.2 Implementation Details. By default we allow the retriever and reranker to take the first 2048 tokens as input without any segmentation, which is a reasonable trade-off between input sequence length and the cost of training. This approach covers about 77% of the documents in the corpus entirely. We create the training data for the document retriever and reranker models based on the 300k training examples in the training set. Similar to the approach for passage ranking, we sample the hard negative documents to train repLLaMA from the top-100 hard negatives from BM25 and our implementation of CoCondenser on the MS MARCO document retrieval training data. Here, BM25 directly indexes the complete documents, while CoCondenser retrieves documents using the aforementioned MaxP strategy. The hard negatives for rankLLaMA are selected from the top-100 results of repLLaMA.

We follow a similar setup as in the passage ranking task to train both *document* repLLaMA and rankLLaMA, with the same

	Train MRR@10	Dev MRR@10	DL19 nDCG@10	DL20 nDCG@10
FT	**46.6**	**41.6**	72.8	69.9
LoRA	40.8	41.2	**74.3**	**72.1**

Table 4: Comparison between full fine-tuning (FT) and LoRA when training repLLaMA for the passage retrieval task.

	4096*	4096	2048	1024	768	512	256
MS PSG Dev (MRR@10)	**41.2**	41.2	41.0	40.4	40.4	39.9	38.7
BEIR-13 Avg (nDCG@10)	**55.1**	55.0	54.6	53.8	53.5	52.6	50.3

Table 5: Comparison of passage ranking effectiveness varying vector dimensionality. 4096* is the effectiveness of default repLLaMA; other results are from the MRL [14] variant.

Retriever	MRR@10	→	Reranker	MRR@10
BM25	18.4		monoT5-3B	39.8
			rankLLaMA-7B	**42.8**
GTR	36.6		RankT5-3B	43.8
			rankLLaMA-7B	**44.8**
repLLaMA	41.2		rankLLaMA-7B	**44.9**

Table 6: Comparison of passage reranking for rankLLaMA when reranking different top-1000 retriever outputs.

computing resources. For the MS MARCO document corpus, this results in a 49G (flat) index after encoding the entire corpus.

3.2.3 Results. Table 3 reports the effectiveness of our repLLaMA–rankLLaMA pipeline for full-document retrieval on the MS MARCO document corpus. We see that both our retriever and reranker outperform existing methods. repLLaMA achieves an MRR@100 score that is about 3 points higher than CoCondenser, while rankLLaMA exceeds (to our knowledge) the current state-of-the-art document reranker, MORES+ [8], by 1 point in MRR@100.

4 ABLATIONS AND ANALYSES

Full Fine-Tuning vs. LoRA. When fine-tuning LLMs, a key decision is whether to perform "full" fine-tuning or to use a parameter-efficient method such as LoRA. Table 4 compares the effectiveness of these two approaches on repLLaMA for the passage retrieval task. We see that full fine-tuning achieves a much higher MRR@10 score than LoRA on the training set; however, this improvement does not translate over to the development set. Interestingly, on the TREC DL19/DL20 datasets, which are derived from independent human judgments, LoRA demonstrates better effectiveness. This suggests that full fine-tuning may be prone to overfitting on the training set distribution, while LoRA, with significantly fewer parameters, can generalizable better. For this reason, all the models presented in our main experiments (Section 3) use LoRA.

Representation Dimensionality. By default, repLLaMA generates 4096 dimensional vectors. We trained a variant of repLLaMA following the Matryoshka Representation Learning (MRL) training strategy [14], which enables repLLaMA to generate representations with flexible dimensionality. As shown in Table 5, the effectiveness of repLLaMA decreases with smaller vectors, but the effectiveness is largely preserved, especially for in-domain evaluation. We observe gradual degradation of effectiveness on BEIR, indicating that repLLaMA has the flexibility to adapt to constrained vector sizes.

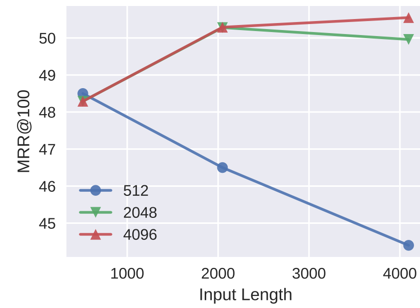

Figure 1: Comparison of document ranking MRR@100 scores for rankLLaMA with different maximum input lengths. Each line represents a model trained with a specific maximum length, while points along the line indicate the effectiveness when varying the input length during inference (reranking).

Reranking Candidates. In Table 6, we show the effectiveness of using rankLLaMA to rerank candidate from retrievers other than repLLaMA. To match the original setting of monoT5 and RankT5, we rerank the top-1000 retriever outputs. Our rankLLaMA outperforms both monoT5 and RankT5 "out of the box", showing the effectiveness of rankLLaMA without relying on repLLaMA.

Input Sequence Length. We investigate the effects of varying the maximum training input length and inference input length on model effectiveness for the document reranking task. Results presented in Figure 1 show a clear trend: the effectiveness of rankLLaMA improves as the maximum training length increases from 512 to 2048, with the MRR@100 score improving from 48.5 to 50.3. When the reranking input length is further increased to 4096, the MRR@100 score rises to 50.6. This demonstrates the model's ability to exploit longer sequences for improved effectiveness.

However, we note that the gains plateau beyond a certain length, suggesting a point of diminishing returns. The MRR@100 for the model trained with a length of 4096 is only 0.3 points higher than the model trained with a length of 2048, when evaluated on input lengths that match their training lengths. Moreover, the model trained with a length of 4096 takes about 8 days to train using 16 × V100 GPUs, while the model with a length of 2048 takes about 4 days. The same relative latency costs apply to inference as well. Therefore, while rankLLaMA can handle much longer input documents, it is crucial to balance this capability with the practical considerations of computational efficiency.

5 CONCLUSION

Our study shows that large language models (LLMs) can be effectively fine-tuned to function as dense retrievers and pointwise rerankers, establishing a point of reference for future multi-stage retrieval systems. This approach surpasses the effectiveness of smaller models, handles longer texts, and highlights the potential for enhancing text retrieval with LLMs.

REFERENCES

[1] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *arXiv:1611.09268* (2016).

[2] 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)*. 1870–1879.

[3] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the TREC 2020 Deep Learning Track. *arXiv:2102.07662* (2021).

[4] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. Overview of the TREC 2019 Deep Learning Track. *arXiv:2003.07820* (2020).

[5] Zhuyun Dai and Jamie Callan. 2019. Deeper Text Understanding for IR with Contextual Neural Language Modeling. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19)*. 985–988.

[6] Tri Dao. 2023. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. *arXiv:2307.08691* (2023).

[7] 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)*. 4171–4186.

[8] Luyu Gao and Jamie Callan. 2022. Long Document Re-Ranking with Modular Re-Ranker. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22)*. 2371–2376.

[9] Luyu Gao and Jamie Callan. 2022. Unsupervised Corpus Aware Language Model Pre-training for Dense Passage Retrieval. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2843–2853.

[10] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. Precise Zero-Shot Dense Retrieval without Relevance Labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1762–1777.

[11] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

[12] Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active Retrieval Augmented Generation. *arXiv:2305.06983* (2023).

[13] 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)*. 6769–6781.

[14] Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, and Ali Farhadi. 2022. Matryoshka Representation Learning. In *Advances in Neural Information Processing Systems*.

[15] 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*. 9459–9474.

[16] Jimmy Lin. 2021. A Proposed Conceptual Framework for a Representational Approach to Information Retrieval. *arXiv:2110.01529* (2021).

[17] 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)*. 2356–2362.

[18] Xueguang Ma, Xinyu Crystina Zhang, Ronak Pradeep, and Jimmy Lin. 2023. Zero-Shot Listwise Document Reranking with a Large Language Model. *arXiv:2305.02156* (2023).

[19] Niklas Muennighoff. 2022. SGPT: GPT Sentence Embeddings for Semantic Search. *arXiv:2202.08904* (2022).

[20] Arvind Neelakantan et al. 2022. Text and Code Embeddings by Contrastive Pre-Training. *arXiv:2201.10005* (2022).

[21] Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernandez Abrego, Ji Ma, Vincent Zhao, Yi Luan, Keith Hall, Ming-Wei Chang, and Yinfei Yang. 2022. Large Dual Encoders Are Generalizable Retrievers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 9844–9855.

[22] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *arXiv:1901.04085* (2019).

[23] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-Stage Document Ranking with BERT. *arXiv:1910.14424* (2019).

[24] OpenAI. 2023. GPT-4 Technical Report. *arXiv:2303.08774* (2023).

[25] 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*. 2523–2544.

[26] Ronak Pradeep, Rodrigo Nogueira, and Jimmy Lin. 2021. The Expando-Mono-Duo Design Pattern for Text Ranking with Pretrained Sequence-to-Sequence Models. *arXiv:2101.05667* (2021).

[27] Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023. RankVicuna: Zero-Shot Listwise Document Reranking with Open-Source Large Language Models. *arXiv:2309.15088* (2023).

[28] Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. 2023. Large Language Models are Effective Text Rankers with Pairwise Ranking Prompting. *arXiv:2306.17563* (2023).

[29] Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. RocketQA: An Optimized Training Approach to Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 5835–5847.

[30] 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. *Journal of Machine Learning Research* 21, 140 (2020), 1–67.

[31] Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen tau Yih. 2023. REPLUG: Retrieval-Augmented Black-Box Language Models. *arXiv:2301.12652* (2023).

[32] Weiwei Sun, Lingyong Yan, Xinyu Ma, Pengjie Ren, Dawei Yin, and Zhaochun Ren. 2023. Is ChatGPT Good at Search? Investigating Large Language Models as Re-Ranking Agent. *arXiv:2304.09542* (2023).

[33] 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 *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

[34] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a Large-scale Dataset for Fact Extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. 809–819.

[35] Hugo Touvron et al. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv:2307.09288* (2023).

[36] Hugo Touvron et al. 2023. LLaMA: Open and Efficient Foundation Language Models. *arXiv:2302.13971* (2023).

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*.

[38] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2023. SimLM: Pre-training with Representation Bottleneck for Dense Passage Retrieval. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2244–2258.

[39] Liang Wang, Nan Yang, and Furu Wei. 2023. Query2doc: Query Expansion with Large Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Singapore, 9414–9423.

[40] Shitao Xiao, Zheng Liu, Yingxia Shao, and Zhao Cao. 2022. RetroMAE: Pre-Training Retrieval-oriented Language Models Via Masked Auto-Encoder. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 538–548.

[41] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. In *International Conference on Learning Representations*.

[42] Nan Yang, Tao Ge, Liang Wang, Binxing Jiao, Daxin Jiang, Linjun Yang, Rangan Majumder, and Furu Wei. 2023. Inference with Reference: Lossless Acceleration of Large Language Models. *arXiv:2304.04487* (2023).

[43] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Z. Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jianyun Nie, and Ji rong Wen. 2023. A Survey of Large Language Models. *arXiv:2303.18223* (2023).

[44] Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. 2023. RankT5: Fine-Tuning T5 for Text Ranking with Ranking Losses. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23)*. 2308–2313.