Fine-Tuning LLaMA for Multi-Stage Text Retrieval

Xueguang Ma † **Liang Wang** ‡ **Nan Yang** ‡ **Furu Wei** ‡ **Jimmy Lin** †

† David R. Cheriton School of Computer Science, University of Waterloo
‡ Microsoft Research

Abstract

The effectiveness of multi-stage text retrieval has been solidly demonstrated since before the era of pre-trained language models. However, most existing studies utilize models that predate recent advances in large language models (LLMs). This study seeks to explore potential improvements that state-of-the-art LLMs can bring. We conduct a comprehensive study, fine-tuning the latest LLaMA model both as a dense retriever (RepLLaMA) and as a pointwise reranker (RankLLaMA) for both passage retrieval and document retrieval using the MS MARCO datasets. Our findings demonstrate that the effectiveness of large language models indeed surpasses that of smaller models. Additionally, since LLMs can inherently handle longer contexts, they can represent entire documents holistically, obviating the need for traditional segmenting and pooling strategies. Furthermore, evaluations on BEIR demonstrate that our RepLLaMA–RankLLaMA pipeline exhibits strong zero-shot effectiveness. Model checkpoints from this study are available on HuggingFace.[1]

1 Introduction

Text retrieval, which entails identifying and ranking the most relevant documents or text snippets in response to a query, is crucial in various open-domain language comprehension tasks (Petroni et al., 2021), including web search (Bajaj et al., 2016), open-domain question answering (Chen et al., 2017), and fact verification (Thorne et al., 2018). Retrieval also plays an important role in enhancing the effectiveness of large language models (LLMs) in a retrieval-augmented generation (RAG) pipeline (Lewis et al., 2020b; Shi et al., 2023). This approach not only mitigates hallucinations but also enables LLMs to access knowledge that is not captured within their parameters (Yang et al., 2023; Jiang et al., 2023).

[1] https://huggingface.co/castorini

A typical multi-stage text retrieval pipeline consists of a *retriever*, designed to efficiently locate the top-k relevant texts from a corpus, and a *reranker*, which further refines the order of the retrieved candidates to improve output quality (Nogueira and Cho, 2019). Both retrievers and rerankers have significantly benefited from the advent of pre-trained language models based on Transformers (Vaswani et al., 2017) such as BERT (Devlin et al., 2019) and T5 (Raffel et al., 2020). These models are trained to encode queries and documents into vector representations for retrieval (Karpukhin et al., 2020; Lin, 2021) or to directly score the relevance between a query and a document for reranking (Nogueira et al., 2019; Zhuang et al., 2023).

Recent large language models with billions of parameters, fine-tuned to follow instructions, such as InstructGPT (Ouyang et al., 2022), GPT-4 (OpenAI, 2023), and LLaMA (Touvron et al., 2023a,b), have exhibited extraordinary capabilities in many NLP tasks, surpassing previous smaller pre-trained language models (Zhao et al., 2023). For retrieval, recent methods such as LRL (Ma et al., 2023), RankGPT (Sun et al., 2023), and PRP (Qin et al., 2023) have explored prompting LLMs to perform zero-shot reranking using pairwise or listwise approaches. These methods leverage LLMs by viewing reranking as text generation.

However, we see a number of potential issues. First, these methods do not address the entire multi-stage pipeline, as it is challenging to cast retrieval from a large corpus as a text generation task. Second, they do not leverage labeled data when available. Finally, these rerankers are not efficient because they do not support parallel scoring and are slowed by their multi-pass decoding design.

Therefore, we argue that fine-tuning state-of-the-art large language models to function as retrievers and rerankers can yield better effectiveness than previous smaller models. This approach can also optimally utilize LLMs within multi-stage

pipelines. Thus, we are motivated to investigate the following research question: How do state-of-the-art large language models perform when specifically fine-tuned for multi-stage text retrieval?

Our study aims to answer this question by conducting a comprehensive investigation into fine-tuning the latest LLaMA-2 model (Touvron et al., 2023b), a state-of-the-art, open-source large language model, as both a retriever and a reranker, which we refer to as RepLLaMA and RankLLaMA, respectively. Specifically, we utilize the MS MARCO (Bajaj et al., 2016) and BEIR (Thakur et al., 2021) datasets for our experiments. Our findings suggest that large language models surpass previous smaller models, achieving state-of-the-art effectiveness for both retrieval and reranking through a straightforward training regime and exhibiting strong zero-shot effectiveness. Furthermore, we observe that LLMs, which are inherently pre-trained on longer contexts, demonstrate potential in representing entire documents, thereby eliminating the need for traditional segmenting and pooling strategies for document retrieval.

2 Method

2.1 Preliminaries

Task Definition Given a query Q and a corpus $C = \{D_1, D_2, ..., D_n\}$ consisting of n documents, the goal of text retrieval is to find the k documents that are most relevant to the query Q, with $k \ll n$. In a multi-stage retrieval pipeline composed by a retriever and a reranker, the retriever's task is to efficiently generate the top-k candidates that are relevant to the query based on the similarity metric $\text{Sim}(Q, D) \in \mathbb{R}$. The reranker's task is to reorder these k candidate documents further to improve the relevance order using a more effective, but typically more computationally expensive reranking model. Note that "document" in this context can refer to an arbitrary information snippet, including sentences, passages, or full documents. While a multi-stage pipeline can contain multiple rerankers, in this paper we focus on a single reranker.

Modern retrievers typically follow a bi-encoder architecture that encodes text into vector representations, with $\text{Sim}(Q, D)$ computed as the dot product of the vector representations of the query Q and a document D (Karpukhin et al., 2020). In contrast, a (pointwise) reranker typically takes both the query and a candidate document as input to directly generate a relevance score. These scores are then used to reorder the candidates (Nogueira et al., 2019; Gao et al., 2021).

LLaMA LLaMA (Touvron et al., 2023a) is an auto-regressive, decoder-only large language model based on the Transformer architecture. The model is characterized by its billions of parameters, pre-trained on a vast amount of web data. Being uni-directional means that the model's attention mechanism only considers the preceding elements in the input sequence when making predictions. Specifically, given an input sequence $x = [t_1, t_2, ..., t_{n-1}]$, the model computes the probability of the next token t_n based solely on the preceding tokens. The prediction process can be mathematically represented as $P(t_n|t_1, ..., t_{n-1})$, where P denotes the probability and t_n represents the next element in the sequence.

2.2 Retriever

Our retriever model, called RepLLaMA, follows the bi-encoder dense retriever architecture proposed in DPR (Karpukhin et al., 2020), but with the backbone model initialized with LLaMA.

Previous work on dense retriever models often uses a bi-directional encoder-only model like BERT, taking the representation of the prepended `[CLS]` token as the dense representation of the text input. However, as LLaMA is uni-directional, we append an end-of-sequence token `</s>` to the input query or document to form the input sequence to LLaMA. Thus, the vector embedding of a query or a document is computed as:

$$V_T = \text{Decoder}(\text{`}t_1 \text{␣} t_2 \text{␣} ... \text{␣} t_k \texttt{</s>}\text{'})[-1]$$

where $\text{Decoder}(\cdot)$ represents the LLaMA model, which returns the last layer token representation for each input token. We take the representation of the end-of-sequence token as the representation of the input sequence $t_1 \dots t_k$, which can be 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 representation 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:

$$\mathcal{L}(Q, D^+, \{D_\text{N}\}) = -\log p(D = D^+ \mid Q) = -\log \frac{\exp(\text{Sim}(Q, D^+))}{\exp(\text{Sim}(Q, D^+)) + \sum_{D_i^- \in \{D_\text{N}\}} \exp(\text{Sim}(Q, D_i^-))}$$

Here, D^+ represents a document that is relevant to the query Q (based on human labels), while $\{D_N\}$

denotes a set of documents that is not relevant to the query. The set of negative documents 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.

During the inference phase, the query is typically encoded in real-time and the top-k similar documents are searched within the pre-encoded corpus using an efficient approximate nearest neighbour search library such as HNSW (Malkov and Yashunin, 2020). However, in this study, we opt to perform exact nearest neighbour search using flat indexes to evaluate model effectiveness.

2.3 Reranker

Our reranker model, referred to as RankLLaMA, 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 (Nogueira et al., 2019).

In more detail, RankLLaMA reranks a query–document pair as follows:

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

where Linear($\cdot$) is a linear projection layer that projects the last layer representation of the end-of-sequence token to a scalar. Similar to the retriever, the model is optimized by contrastive loss. However, 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.

During the inference stage, the top candidate documents retrieved by RepLLaMA are reordered. This reordering is based on the relevance score that RankLLaMA assigns to each query–document pair, with the documents arranged in descending order of relevance.

3 Experiments

We conduct experiments on MS MARCO passage ranking and document ranking datasets to investigate the effectiveness of the multi-stage text retrieval pipeline built using RepLLaMA and RankLLaMA for both passage and document retrieval.

3.1 Passage Retrieval

Dataset We train our retriever and reranker models with LLaMA on the training split of the MS MARCO passage ranking dataset (Bajaj et al., 2016), which consists of approximately 500k training examples. As discussed in Section 2.2, the incorporation of hard negatives is crucial for the effective training of the retriever. In our case, we use a blend of BM25 and CoCondenser (Gao and Callan, 2022b) 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, comprising 6980 queries. Effectiveness is reported using MRR@10 as the metric. In addition, we also evaluate our models on the TREC DL19/DL20 passage ranking test collections (Craswell et al., 2020, 2021), which include 43 and 54 queries, respectively. These collections utilize the same passage corpus as MS MARCO, but provide query sets with dense, graded human relevance judgments. Following standard practice, we adopt nDCG@10 as the evaluation metric in our experiments.

In addition, we assess the zero-shot effectiveness of RepLLaMA and RankLLaMA on BEIR (Thakur et al., 2021), which is a compilation of 18 datasets that spans a variety of domains (e.g., news, medical) and retrieval tasks (e.g., fact verification, question answering). We focus our evaluation on the 13 datasets that are publicly available.

Implementation Details We initialize our models with the LLaMA-2-7B checkpoint[2] and train on 16 $\times$ 32G V100 GPUs. For RepLLaMA, we extract the final layer representation of the </s> token as the dense representation, which has a dimensionality of 4096. Additionally, we normalize these dense representations into unit vectors during

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

	Model size	Source prev.	top-k	DEV MRR@10	DEV R@1k	DL19 nDCG@10	DL20 nDCG@10
Retrieval							
BM25 (Lin et al., 2021)	-	-	$\|C\|$	18.4	85.3	50.6	48.0
ANCE (Xiong et al., 2021)	125M	-	$\|C\|$	33.0	95.9	64.5	64.6
CoCondenser (Gao and Callan, 2022b)	110M	-	$\|C\|$	38.2	98.4	71.7	68.4
GTR-base (Ni et al., 2022)	110M	-	$\|C\|$	36.6	98.3	-	-
GTR-XXL (Ni et al., 2022)	4.8B	-	$\|C\|$	38.8	99.0	-	-
OpenAI Ada2 (Neelakantan et al., 2022)	?	-	$\|C\|$	34.4	98.6	70.4	67.6
bi-SimLM (Wang et al., 2023)	110M	-	$\|C\|$	39.1	98.6	69.8	69.2
RepLLaMA	7B	-	$\|C\|$	**41.2**	**99.4**	**74.3**	**72.1**
Reranking							
monoBERT (Nogueira et al., 2019)	110M	BM25	1000	37.2	85.3	72.3	72.2
cross-SimLM (Wang et al., 2023)	110M	bi-SimLM	200	43.7	98.7	74.6	72.7
RankT5 (Zhuang et al., 2023)	220M	GTR	1000	43.4	98.3	-	-
RankLLaMA	7B	RepLLaMA	200	44.9	99.4	75.6	77.4
RankLLaMA-13B	13B	RepLLaMA	200	**45.2**	**99.4**	**76.0**	**77.9**
RankVicuna (Pradeep et al., 2023)	7B	BM25	100	-	-	66.8	65.5
PRP (Qin et al., 2023)	20B	BM25	100	-	-	72.7	70.5
RankGPT$_{3.5}$ (Sun et al., 2023)	?	BM25	100	-	-	65.8	72.9
RankGPT$_4$ (Sun et al., 2023)	?	RankGPT$_{3.5}$	30	-	-	75.6	70.6

Table 1: The effectiveness of RepLLaMA and RankLLaMA on the MS MARCO passage corpus compared to existing methods. For the retriever, we compare against models trained with binary human judgments, without distillation from a reranker. Evaluation figures are copied from the original papers except for OpenAI Ada2, which is the successor to cpt-text (Neelakantan et al., 2022) and available as a commercial API. The effectiveness numbers of Ada2 are taken from Lin et al. (2023).

both the training and inference stages, ensuring that their L2-norms are equal to 1. After pre-encoding the entire corpus, we end up with a 135G flat index for brute-force search.

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 recent memory efficiency solutions, including LoRA (Hu et al., 2022), flash attention (Dao, 2023), 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 initialization.[3]

In-Domain Evaluation Table 1 presents the effectiveness of RepLLaMA and RankLLaMA on the MS MARCO passage corpus in comparison to existing methods.

For retrieval, RepLLaMA outperforms all competing methods, achieving the highest effectiveness. The closest system in terms of effectiveness is bi-SimLM (Wang et al., 2023), which RepLLaMA outperforms by 2 points MRR@10 on the dev queries. However, bi-SimLM involves a pre-training stage for enhancing the text representation. In contrast, RankLLaMA directly uses the off-the-shelf LLaMA model as initialization. When compared to the GTR-XXL retriever, which also uses a model with billions of parameters based on the T5-encoder (Ni et al., 2022), our model achieves higher MRR@10 and Recall@1k on the dev queries and on TREC DL19/DL20. Specifically, RepLLaMA achieves 2.4 points higher MRR@10 and 0.4 points higher Recall@1k than GTR-XXL.

It is worth noting that recent studies have shown the potential to further improve dense retrieval models by learning from soft labels provided by a reranker via optimizing KL-divergence. However, in this study, we train our model with only binary judgments. Training RepLLaMA by knowledge distillation will likely lead to further improvements, but we save this for future work.

For reranking, RankLLaMA reranks the top-200 passages from RepLLaMA, resulting in the highest end-to-end effectiveness of any multi-stage re-

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

	BM25	GTR-XXL	cpt-text-XL	Ada2	SGPT	RepLLaMA	RankT5	RankLLaMA	RankLLaMA
model size	-	4.8B	175B	?	5.8B	7B	220M	7B	13B
add. pretrain	-	Y	Y	?	Y	N	-	-	-
Arguana	39.7	54.0	43.5	56.7	51.4	48.6	33.0	56.0	50.8
Climate-FEVER	16.5	26.7	22.3	23.7	30.5	31.0	21.5	28.0	29.2
DBPedia	31.8	40.8	43.2	40.2	39.9	43.7	44.2	48.3	48.7
FEVER	65.1	74.0	77.5	77.3	78.3	83.4	83.2	83.9	86.2
FiQA	23.6	46.7	51.2	41.1	37.2	45.8	44.5	46.5	48.1
HotpotQA	63.3	59.9	68.8	65.4	59.3	68.5	71.0	75.3	76.4
NFCorpus	32.2	34.2	40.7	35.8	36.2	37.8	38.1	30.3	28.4
NQ	30.6	56.8	-	48.2	52.4	62.4	61.4	66.3	66.7
Quora	78.9	89.2	63.8	87.6	84.6	86.8	83.1	85.0	81.7
SCIDOCS	14.9	16.1	-	18.6	19.7	18.1	18.1	17.8	19.1
SciFact	67.9	66.2	75.4	73.6	74.7	75.6	75.0	73.2	73.0
TREC-COVID	59.5	50.1	64.9	81.3	87.3	84.7	80.7	85.2	86.1
Touche-2020	44.2	25.6	29.1	28.0	25.4	30.5	44.0	40.1	40.6
Average	43.7	49.3	-	52.1	52.1	**55.1**	53.7	**56.6**	56.5

Table 2: Zero-shot effectiveness of RepLLaMA and RankLLaMA on BEIR datasets. The "*add. pretrain*" row indicates whether the retriever model has undergone additional contrastive pre-training before supervised fine-tuning. The zero-shot effectiveness numbers of Ada2 are taken from Kamalloo et al. (2023).

trieval system that we are aware of. Our complete RepLLaMA–RankLLaMA pipeline beats the previous state-of-the-art reranker, RankT5 (Zhuang et al., 2023), by 1.5 points MRR@10. Furthermore, our RankLLaMA-13B model outperforms the 7B model, achieving 0.3 points higher MRR@10 and slightly higher nDCG@10 on both DL19 and DL20. This indicates the potential for further improvements with even larger models.

Compared to RankGPT$_4$ (Sun et al., 2023), which prompts GPT-4 to perform passage reranking through permutation generation within a multi-stage retrieval pipeline, our RepLLaMA–RankLLaMA pipeline outperforms it by 0.4 and 7.3 nDCG@10 points on DL19 and DL20, respectively. 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.

Zero-Shot Evaluation The zero-shot evaluation of RepLLaMA and RankLLaMA on the BEIR datasets is presented in Table 2. Both models demonstrate superior zero-shot effectiveness, outperforming existing models. RepLLaMA surpasses other existing dense retrievers with billions of parameters. Specifically, it outperforms SGPT (Muennighoff, 2022) and Ada2 by 3 points and exceeds GTR-XXL by approximately 6 points. Note that these methods 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 by an average of 1.5 points on nDCG@10. Interestingly, the larger RankLLaMA-13B model does not appear to yield any further improvements.

3.2 Document Retrieval

Dataset The document retrieval task aims to rank document-length texts, which present the challenge of handling long input sequences (Bajaj et al., 2016). As illustrated in Figure 1, 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 rerankers based on smaller pre-trained language models like BERT (Devlin et al., 2019).

The standard solution to manage long sequences for retrieval is the MaxP strategy (Dai and Callan, 2019), which involves dividing the document into overlapping segments and determining the document relevance score based on the segment with the highest score. However, this process involves a heuristic pooling strategy and runs the risk of losing information spread across long contexts. Recent language models pre-trained on longer sequences (e.g., 4096 tokens for LLaMA-2) offer the potential to represent document-length texts "in one go", reducing the need for segmentation.

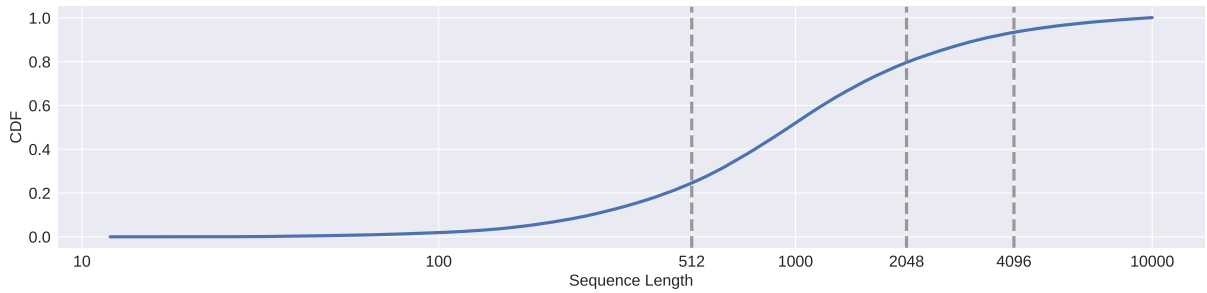

Figure 1: Cumulative distribution function of document lengths in the MS MARCO document corpus, showing the proportion of documents that has a length less than a specific value (determined by the LLaMA tokenizer). For clarity, we exclude 3% of documents with a length exceeding 10,000 tokens.

	Model size	Source prev.	top-k	Seg. Y/N	Dev MRR@100	Dev R@1k	DL19 nDCG@10	DL20 nDCG@10
Retrieval								
BM25 (Lin et al., 2021)	-	-	$\|C\|$	N	23.0	85.3	51.8	52.9
BM25-Q2D (Pradeep et al., 2021)	-	-	$\|C\|$	Y	31.8	94.9	61.2	59.6
CoCondenser-MaxP	110M	-	$\|C\|$	Y	42.5	93.9	64.8	**64.0**
RepLLaMA	7B	-	$\|C\|$	N	**45.6**	**98.9**	**65.0**	63.2
Reranking								
monoT5 (Pradeep et al., 2021)	3B	BM25-Q2D	10000	Y	41.1	94.9	-	-
MORES+ (Gao and Callan, 2022a)	110M	CoCondenser	100	Y	49.3	-	-	-
RankLLaMA	7B	RepLLaMA	100	N	**50.3**	**98.9**	**67.7**	**67.4**

Table 3: The effectiveness of RepLLaMA and RankLLaMA on the MS MARCO document corpus compared to existing methods.

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 in passage ranking, we sample the hard negative documents to train RepLLaMA from the top-100 hard negatives from BM25 and our implementation of CoCondenser-MaxP. Here, BM25 directly indexes the entire documents, while CoCondenser retrieves documents using the aforementioned MaxP strategy. The hard negatives for RankLLaMA are selected from the top-100 results of RepLLaMA.

Evaluation of document retrieval is performed on the development split of the MS MARCO document ranking dataset, which contains 5193 queries. Additionally, we evaluate our models on the TREC DL19/DL20 document ranking tasks, comprising 43 and 45 queries, respectively.

Implementation Details We follow a similar setup as in the passage ranking task to train both *document* RepLLaMA and RankLLaMA, with the same computing resources. However, there are two key differences: First, the models are trained with a batch size of 128, with each query sampling 7 hard negative passages. Second, during inference, RepLLaMA retrieves the top-1000 documents while RankLLaMA reranks the top-100 documents that are retrieved by RepLLaMA. The document model also generates text embeddings with 4096 dimensions. For the MS MARCO document corpus, this results in a 49G (flat) index after pre-encoding the entire corpus.

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 approximately 3 points higher than CoCondenser-MaxP, while RankLLaMA exceeds (to our knowledge) the current state-of-the-art document reranker, MORES+ (Gao and Callan, 2022a), by 1 point in MRR@100.

We again emphasize that both our retriever and reranker do not require document segmentation

	Train	Dev	DL19	DL20
FT	**46.6**	**41.6**	72.8	69.9
LoRA	40.8	41.2	**74.3**	**72.1**

Table 4: Comparison of MRR@10 between full fine-tuning (FT) and LoRA when training RepLLaMA for the passage retrieval task.

and rank score aggregation. Instead, RepLLaMA directly consumes the entire document, and RankLLaMA directly scores the relevance of the entire query–document pair.

4 Ablation Study and Analysis

4.1 Full Fine-Tuning vs. LoRA

When fine-tuning large language models, a key decision is whether to conduct full fine-tuning, which updates all parameters in the model, or to use a parameter-efficient method such as LoRA. Table 4 compares the effectiveness of RepLLaMA when trained with full fine-tuning and LoRA for the passage retrieval task. Both models are trained on the training set for one epoch.

We see that full fine-tuning achieves an MRR@10 score that is approximately 6 points higher than with LoRA on the training set. However, on the development set, full fine-tuning only improves effectiveness by 0.4 points compared to LoRA. 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 instead of full fine-tuning.

4.2 Input Sequence Length

As discussed in Section 3.2, RankLLaMA has the advantage of accommodating longer inputs compared to previous models like BERT since its LLaMA backbone was pre-trained with a longer context window. 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 2 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.

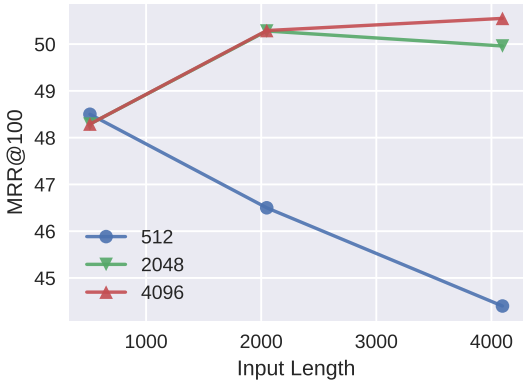

Figure 2: Comparison of document ranking MRR@100 scores for RankLLaMA trained with different maximum input lengths and evaluated using different maximum input lengths. Each line represents a model trained with a specific maximum length, while points along the line indicate the effectiveness when varying the input length during inference (reranking).

However, it is important to 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 $\times$ 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 Related Work

5.1 Large Language Models

Pre-trained language models based on the Transformer architecture (Vaswani et al., 2017) have demonstrated impressive capabilities when fine-tuned for various downstream tasks since the advent of BERT (Devlin et al., 2019). Depending on their architecture, pre-trained Transformers can be classified into three categories: encoder-only models (Devlin et al., 2019; Liu et al., 2019; Conneau et al., 2020), encoder–decoder models (Raffel et al.,

2020; Lewis et al., 2020a), and decoder-only models (Radford et al., 2018). Decoder-only models like GPT/GPT-2 have been lauded for their simplicity in terms of model architecture and pre-training procedures (Radford et al., 2018, 2019).

Recent research has shown that scaling up LLMs by pre-training larger decoder-only models using larger and higher quality corpora can significantly enhance model capabilities for general-purpose NLP tasks such as question answering and code generation (Wei et al., 2022; Chen et al., 2021). This is achieved by fine-tuning the pre-trained LLMs with instruction-following data using reinforcement learning with human feedback. InstructGPT (Ouyang et al., 2022) and GPT-4 (OpenAI, 2023) are two popular representatives in this class of models. Among the many implementations of open-source large language models, LLaMA (Touvron et al., 2023a,b) is among the most recent and among the top-performing on a variety of tasks.

5.2 Multi-Stage Text Retrieval

While multi-stage retrieval pipelines date back well over a decade (Matveeva et al., 2006; Cambazoglu et al., 2010; Wang et al., 2011), they have benefited immensely from pre-trained language models such as BERT in recent years, starting with the monoBERT reranking model (Nogueira and Cho, 2019). Nogueira et al. (2019) proposed a multi-stage retrieval pipeline that employs a BM25 retriever followed by two BERT-based reranking stages. This design demonstrates the effectiveness of pre-trained language models in reranking tasks. RankLLaMA follows the same basic design as monoBERT. The dense passage retriever (DPR) further proposed to fine-tune BERT to replace the BM25 retriever with a dense retrieval model in a bi-encoder design (Karpukhin et al., 2020). DPR encodes text into low-dimensional dense vector representations and treats retrieval as a nearest-neighbor search task. RepLLaMA follows the same bi-encoder design.

Several solutions have been introduced to enhance the effectiveness of retrievers and rerankers in a multi-stage pipeline. On the retriever side, works such as ANCE (Xiong et al., 2021), RocketQA (Qu et al., 2021), CoCondenser (Gao and Callan, 2022b), RetroMAE (Xiao et al., 2022), and SimLM (Wang et al., 2023), have shown that augmenting the training data with hard negative mining or continuous retrieval-oriented pre-training can improve the effectiveness of dense retrievers. On the reranker side, monoT5 (Nogueira et al., 2020) and monoELECTRA (Pradeep et al., 2022) demonstrated that initializing the reranker with a custom pre-trained model can enhance effectiveness. Gao et al., 2021 proposed using a contrastive loss for reranker training to replace the default pairwise loss. Zhuang et al. (2023) studied the use of T5 as a reranker, analyzing the influence of different model architectures and loss functions. However, directly fine-tuning modern billion-parameter-scale large language models for multi-stage retrieval has not been explored to date.

Recently, LLMs have demonstrated impressive effectiveness when prompted to perform few-shot or zero-shot text generation. As mentioned in the introduction, researchers have cast reranking as text generation. These models can be leveraged to directly generate a reordered list of candidates, e.g., LRL (Ma et al., 2023), RankGPT (Sun et al., 2023), RankVicuna (Pradeep et al., 2023). Alternatively, they can compare passages in a pairwise manner, e.g., PRP (Qin et al., 2023). Although prompt-based methods have shown good zero-shot effectiveness, they require multiple decoding passes, thus making them slow and non-parallelizable. Furthermore, reranking with prompts makes it difficult to exploit available human judgments such as MS MARCO (Bajaj et al., 2016) to further improve effectiveness. Finally, these approaches do not allow for joint reranker–retriever optimization. In contrast, we address all these issues.

Our work is most similar to GPT-XXL (Ni et al., 2022) and SGPT (Muennighoff, 2022), which also used billion-parameter-scale models as backbones of dense retrievers, achieving better zero-shot effectiveness than smaller models. However, LLaMA has demonstrated even better effectiveness on natural language generation tasks, suggesting that it might serve as a better backbone and warranting further exploration. The cpt-text model (Neelakantan et al., 2022), initialized with the 175-billion-parameter GPT-3 model, also shows strong zero-shot effectiveness. However, cpt-text is not an open-source model. Additionally, none of the models referenced above are fully optimized for a multi-stage retrieval pipeline. Our RepLLaMA and RankLLaMA models are fully open-source and optimized for multi-stage retrieval, achieving state-of-the-art effectiveness on both retrieval and reranking, for both in-domain and out-of-domain evaluations.

6 Conclusion

The successful application of large language models in generative tasks has sparked interest in their potential to enhance retrieval. In this study, we demonstrate that it is possible to fine-tune a large model to act as a dense retriever (RepLLaMA) and a pointwise reranker (RankLLaMA), thereby establishing an effective, state-of-the-art multi-stage retrieval system that outperforms smaller models built on the same basic design. Moreover, our approach offers greater optimization and efficient inference potential than recent methods that prompt large language models for text reranking in a generative manner. This work underscores the potential of leveraging LLMs for retrieval tasks in the future, which we continue to explore.

Acknowledgments

This research was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

References

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*.

B. Barla Cambazoglu, Hugo Zaragoza, Olivier Chapelle, Jiang Chen, Ciya Liao, Zhaohui Zheng, and Jon Degenhardt. 2010. Early exit optimizations for additive machine learned ranking systems. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, page 411–420, New York, NY, USA. Association for Computing Machinery.

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.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. *arXiv:2107.03374*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the TREC 2020 deep learning track. *arXiv:2102.07662*.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. Overview of the TREC 2019 deep learning track. *arXiv:2003.07820*.

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, page 985–988, New York, NY, USA. Association for Computing Machinery.

Tri Dao. 2023. FlashAttention-2: Faster attention with better parallelism and work partitioning. *arXiv:2307.08691*.

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.

Luyu Gao and Jamie Callan. 2022a. 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, page 2371–2376, New York, NY, USA. Association for Computing Machinery.

Luyu Gao and Jamie Callan. 2022b. 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), pages 2843–2853, Dublin, Ireland. Association for Computational Linguistics.

Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021. Rethink training of BERT rerankers in multi-stage retrieval pipeline. In *Advances in Information Retrieval: 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 – April 1, 2021, Proceedings, Part II*, page 280–286, Berlin, Heidelberg. Springer-Verlag.

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*.

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*.

Ehsan Kamalloo, Xinyu Zhang, Odunayo Ogundepo, Nandan Thakur, David Alfonso-hermelo, Mehdi Rezagholizadeh, and Jimmy Lin. 2023. Evaluating embedding APIs for information retrieval. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 518–526, Toronto, Canada. Association for Computational Linguistics.

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.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

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. 2020b. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Jimmy Lin. 2021. A proposed conceptual framework for a representational approach to information retrieval. *arXiv:2110.01529*.

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.

Jimmy Lin, Ronak Pradeep, Tommaso Teofili, and Jasper Xian. 2023. Vector search with OpenAI embeddings: Lucene is all you need. *arXiv:2308.14963*.

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. *arXiv:1907.11692*.

Xueguang Ma, Xinyu Crystina Zhang, Ronak Pradeep, and Jimmy Lin. 2023. Zero-shot listwise document reranking with a large language model. *arXiv:2305.02156*.

Yu A. Malkov and D. A. Yashunin. 2020. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836.

Irina Matveeva, Chris Burges, Timo Burkard, Andy Laucius, and Leon Wong. 2006. High accuracy retrieval with multiple nested ranker. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, page 437–444, New York, NY, USA. Association for Computing Machinery.

Niklas Muennighoff. 2022. SGPT: GPT sentence embeddings for semantic search. *arXiv:2202.08904*.

Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, Johannes Heidecke, Pranav Shyam, Boris Power, Tyna Eloundou Nekoul, Girish Sastry, Gretchen Krueger, David Schnurr, Felipe Petroski Such, Kenny Hsu, Madeleine Thompson, Tabarak Khan, Toki Sherbakov, Joanne Jang, Peter Welinder, and Lilian Weng. 2022. Text and code embeddings by contrastive pre-training. *arXiv:2201.10005*.

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*, pages 9844–9855, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with BERT. *arXiv:1901.04085*.

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*, pages 708–718, Online. Association for Computational Linguistics.

Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-stage document ranking with BERT. *arXiv:1910.14424*.

OpenAI. 2023. GPT-4 technical report. *arXiv:2303.08774*.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke E. Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Francis Christiano, Jan Leike, and Ryan J. Lowe. 2022. Training language models to follow instructions with human feedback. *arXiv:2203.02155*.

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.

Ronak Pradeep, Yuqi Liu, Xinyu Zhang, Yilin Li, Andrew Yates, and Jimmy Lin. 2022. Squeezing water from a stone: A bag of tricks for further improving cross-encoder effectiveness for reranking. In *Advances in Information Retrieval*, pages 655–670, Cham. Springer International Publishing.

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*.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023. RankVicuna: Zero-shot listwise document reranking with open-source large language models. *arXiv:2309.15088*.

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*.

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*, pages 5835–5847, Online. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 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.

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):1–67.

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*.

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*.

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)*.

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)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. LLaMA: Open and efficient foundation language models. *arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov,

Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv:2307.09288*.

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*, volume 30. Curran Associates, Inc.

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)*, pages 2244–2258, Toronto, Canada. Association for Computational Linguistics.

Lidan Wang, Jimmy Lin, and Donald Metzler. 2011. A cascade ranking model for efficient ranked retrieval. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, page 105–114, New York, NY, USA. Association for Computing Machinery.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Huai hsin Chi, F. Xia, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *arXiv:2201.11903*.

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*, pages 538–548, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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 *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

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*.

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*.

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, page 2308–2313, New York, NY, USA. Association for Computing Machinery.