

# Unleashing the Power of LLMs in Dense Retrieval with Query Likelihood Modeling

Hengran Zhang, Keping Bi  
 CAS Key Lab of Network Data  
 Science and Technology, ICT, CAS  
 University of Chinese Academy of  
 Sciences  
 Beijing, China  
 zhanghengran22z@ict.ac.cn  
 bikeping@ict.ac.cn

Shihao Liu, Daiting Shi  
 Baidu Inc  
 Beijing, China  
 liushihao02@baidu.com  
 shidaiting01@baidu.com

Jiafeng Guo  
 CAS Key Lab of Network Data  
 Science and Technology, ICT, CAS  
 University of Chinese Academy of  
 Sciences  
 Beijing, China  
 guojiafeng@ict.ac.cn

Dawei Yin  
 Baidu Inc  
 Beijing, China  
 yindawei@acm.org

Xiaojie Sun  
 CAS Key Lab of Network Data  
 Science and Technology, ICT, CAS  
 University of Chinese Academy of  
 Sciences  
 Beijing, China  
 sunxiaojie@ict.ac.cn

Xueqi Cheng  
 CAS Key Lab of Network Data  
 Science and Technology, ICT, CAS  
 University of Chinese Academy of  
 Sciences  
 Beijing, China  
 cxq@ict.ac.cn

## Abstract

Dense retrieval is a crucial task in Information Retrieval (IR) and is the foundation for downstream tasks such as re-ranking. Recently, large language models (LLMs) have shown compelling semantic understanding capabilities and are appealing to researchers studying dense retrieval. LLMs, as decoder-style generative models, are competent at language generation while falling short on modeling global information due to the lack of attention to tokens afterward. Inspired by the classical word-based language modeling approach for IR, i.e., the query likelihood (QL) model, we seek to sufficiently utilize LLMs' generative ability by QL maximization. However, instead of ranking documents with QL estimation, we introduce an auxiliary task of QL maximization to yield a better backbone for contrastively learning a discriminative retriever. We name our model as LLM-QL. To condense global document semantics to a single vector during QL modeling, LLM-QL has two major components, **Attention Stop (AS)** and **Input Corruption (IC)**. AS stops the attention of predictive tokens to previous tokens until the ending token of the document. IC masks a portion of tokens in the input documents during prediction. Experiments on MSMARCO show that LLM-QL can achieve significantly better performance than other LLM-based retrievers and using QL estimated by LLM-QL for ranking outperforms word-based QL by a large margin. Our code can be found at <https://github.com/Trustworthy-Information-Access/llm-ql>.

---

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.  
 ACM ISBN 978-1-4503-XXXX-X/18/06  
<https://doi.org/XXXXXXX.XXXXXXX>

## CCS Concepts

- Information systems → Language models; Learning to rank; Novelty in information retrieval.

## Keywords

LLMs for dense retrieval, Query likelihood model

## ACM Reference Format:

Hengran Zhang, Keping Bi, Jiafeng Guo, Xiaojie Sun, Shihao Liu, Daiting Shi, Dawei Yin, and Xueqi Cheng. 2018. Unleashing the Power of LLMs in Dense Retrieval with Query Likelihood Modeling. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 12 pages. <https://doi.org/XXXXXXX.XXXXXXX>

## 1 Introduction

Information retrieval (IR) contains two primary stages: retrieval and reranking [13, 67]. Retrieval serves as the cornerstone of information retrieval and is critically important. Its objective is to retrieve relevant passages from a large-scale corpus in response to a query, thereby providing candidate passages for the subsequent reranking stage. In the early stages of retrieval, the focus was primarily on lexical matching between the terms of the query and the passage, with methods such as BM25 [34] and query likelihood (QL) model [46]. However, with the advent of pre-trained language models (PLMs) such as BERT [11], representing passages or queries as dense vectors has gradually become the mainstream approach. These methods typically employ two separate encoders to represent the query and passage and are referred to as dual-encoder models.

Large language models (LLMs) are being widely applied across various fields [27, 54, 61], and garnering increasing attention for their application in retrieval tasks [25, 39, 51]. Unlike bidirectional attention mechanisms in encoder-style pre-trained language models (PLMs) such as BERT, LLMs are typically decoder-style models

that employ unidirectional attention. The task of next token prediction allows LLMs to ingest large amounts of various types of data and thus gain more powerful semantic understanding capability. However, the unidirectional attention during modeling may lead to insufficient representation of global semantics, which is inferior to encoder-style PLMs. Although leveraging the superior semantic understanding ability of LLMs for retrieval looks appealing, it is challenging to do so. Recent studies have attempted to repeat passages as input during encoding [51], to use a bidirectional attention mechanism for encoding during relevance matching fine-tuning [4], or pre-train LLMs to strengthen the global semantic representation capabilities [33].

Since LLMs are decoder-based language models (LMs), it is natural to think of adapting them to retrieval according to the classical language modeling approach to IR [46], i.e., modeling the query likelihood (QL) given the LM of a document. Recent research has also explored modeling ranking with query likelihood using deep generative models such as encoder-decoder-style PLMs (e.g., T5) and appending an LSTM or transformer decoder to BERT [32, 69]. However, although their performance is better than the word-based QL model, they are much worse than BERT-based retrievers. It is not surprising since a generative approach may not perform as well as discriminative models on ranking tasks due to its inability to capture multi-grade relevance and lack of contrastive learning. Moreover, ranking with QL estimation by PLMs is cost-prohibitive to be used for retrieval from a large-scale corpus. Then, **do we have an effective way to leverage the generation capabilities of LLMs and unleash their potential in document representations for retrieval?**

To this end, we propose LLM-QL, which aims to utilize LLMs' generation capabilities for dense retrieval. Instead of modeling relevance matching with a generation process as in [32, 69], we still employ discriminative modeling with a dual encoder as well as contrastive learning and incorporate query likelihood modeling as an auxiliary training task. In this way, the generalization ability of LLMs can be utilized during maximizing query likelihood, which also acts as a better foundation for contrastively learning relevance matching.

As we know, a potent encoder for dense retrieval should be able to 1) sufficiently condense the semantics of documents or passages to a single vector, and 2) capture the potential query needs they may satisfy. We leverage query likelihood modeling to achieve the latter and for the first capability, we propose two strategies in LLM-QL to enhance it: **Attention Stop (AS)** and **Input Corruption (IC)**. Specifically, AS means that during query generation modeling, we stop the attention at the ending token of a document, so that the document semantics are forced to be compressed to this single token for predicting the query. IC means corrupting the document by randomly masking a portion of tokens, aiming to condense as many as possible document semantics to the final representation and improve training efficiency. IC is inspired by Chen [8] for paragraph vector training, where predicting the query with different corrupt portions helps maintain global document semantics sufficiently.

We compare LLM-QL with various baselines on the widely-used MSMACRO [3] dataset and TREC DL 19 and 20 [10], especially retrievers also based on LLMs. Since our approach does not involve any large-scale pre-training, we focus on the comparisons with

methods alike. Experimental results show that LLM-QL significantly outperforms LLMs-based baselines in terms of MRR and NDCG at different cutoffs. We also conduct comprehensive analyses of the components of LLMs, other options of the model, and the ranking performance of query likelihood estimation based on LLM-QL. Our model provides a feasible and promising direction for enhancing LLMs for dense retrieval.

## 2 Related Work

The primary issue in information retrieval is determining the relevance of a document in response to a specific query [17, 17, 22]. Formally, given a query  $q$  and a document  $d$ , the degree of relevance is typically measured using a scoring function based on the query and document representations:

$$\text{Relevance}(q, d) = F(\phi(q), \phi(d)), \quad (1)$$

where  $\phi$  is a function to map the query or the document to a representation vector, and  $F$  is the scoring function based on the interactions between them.  $F$  has two types: (1) a relatively simple matching function, e.g., cosine similarity function, dot product. (2) complex mechanism, e.g., ColBERT [29], I3 Retriever [13]. For the function  $\phi$  of the embedding model, the backbone architectures, to index the query or the document in a low-dimensional and continuous space, are evolving in tandem as the natural language processing (NLP) field advances. In this section, we introduce two backbone architectures used to embed low-dimensional vectors: pre-trained language models (PLMs) based IR, and large language models (LLMs) based IR.

### 2.1 Pre-trained Language Models-Based IR

Pre-trained language models, e.g., BERT [11], RoBERTa [36], and XL-NET [64], have shown superior performance in various NLP tasks. Dense Passage Retrieval (DPR) [28] firstly used the pre-trained language models to encode the representation vector of queries or documents. There are also many subsequent studies to improve DPR to improve the performance of PLM encoding queries and documents, thereby improving retrieval performance. Methods to improve encoding capabilities mainly improved from two aspects: (1) designing pre-training tasks tailored for information retrieval, due to PLMs were pre-trained by token-level tasks, like MLM and Seq2Seq, not for sentence-level or paragraph-level tasks in information retrieval [38]. RetroMAE [60] and SimLM [58] proposed the pre-training methods for IR based on masked auto-encoder(MAE), introducing a shallow decoder to recover the original input based on the sentence embedding and masked input tokens. Although our method also masks the input, unlike MAE, which reconstructs the input, our method predicts the potential query requirements of the document. Chang et al. [7] and Izacard et al. [23] proposed paragraph-level pre-training tasks, e.g., Inverse Cloze Task (ICT). (2) designing effective fine-tuning training methods, mainly focusing on hard negative sampling and training function. ANCE [62] leveraged an Approximate Nearest Neighbor (ANN) index of the corpus to dynamically update and select realistic negative training instances during the learning process. RocketQA [47] proposed cross-batch negatives and denoised hard negatives to improve retrieval performance. There are mainly two types of training functions: contrastive learning and distillation learning. STAR [67] and

ADORE [67] used random negative and static hard negative sampling methods with a dynamic one to optimize retrieval models. Many empirical studies [13, 37, 49, 50, 68] have shown that using ranking models with fine-grained interactions between query  $q$  and document  $d$  (i.e., cross-encoder) guiding retrieval models (i.e., dual-encoder) can help improve retrieval performance. However, distillation learning requires the introduction of additional models, and the training cost is relatively higher than contrastive learning.

## 2.2 Large Language Models-Based IR

Large language models (LLMs), e.g., LLaMA [14, 54], Mistral [24], Qwen [2] and GPT-4 [1], have achieved excellent performance in various fields, especially in various subtasks of natural language generation (NLG). In the era of Pre-trained Language Models (PLMs), pre-trained language models have largely surpassed traditional sparse vector space models and neural network-based retrievers. Moreover, many studies [15, 23, 44, 59] found that with the expansion of the model and training scale, the performance and generality of the PLM-based dense retrieval can be further improved. It is not difficult to infer whether LLM-based retrievers also have the potential to surpass PLM-based retrievers. Recently, many studies [4, 31, 33, 39, 41, 51] have applied decoder-only architecture LLMs to dense retrieval to encode query and document embeddings as representation vectors. For example, RepLLaMA [39] directly replaces the PLMs with LLMs to encode the representation of query and document, and empirical evidence shows that LLM-based retrievers trained using a simple fine-tuning strategy outperform LM-based retrievers trained using complex training strategies. LLM2VEC [4, 31] replaced the decoder-only unidirectional attention with bidirectional attention, further improving retrieval performance. Llama2Vec [33] introduced two pre-training tasks (i.e., EBAE Embedding-based Auto-Encoding and Embedding-based Auto-Regression) for LLMs to adapt LLMs properly so that they can be effectively initialized as the backbone encoder for dense retrieval. Echo [51] repeated the input twice in context and extracted embeddings from the second occurrence to address an architectural limitation of autoregressive models. Unlike these works, in this work, we propose a novel approach that simply introduces generative training before contrastive learning can significantly enhance the retrieval performance of LLMs.

## 3 Preliminaries

### 3.1 Problem Definition

In the current information retrieval systems, the “retriever-then-ranker” pipeline architecture is prevalent. In this paper, we focus on the passage retrieval component of this architecture. Given a query  $q$  and a corpus  $C$  with numerous passages, the retrieval aims to recall as many potentially relevant passages as possible. Formally, given a annotated training dataset  $D = \{(q_i, D_i^+)\}_{i=1}^N$ , where  $q_i$  is the web query,  $D_i^+$  is the relevant passage set for  $q_i$ , and  $N$  is the number of the queries. The core motivation of retrieval is to train a model that can give a higher score for a relevant passage  $d^+ \in D_i^+$  compared to an irrelevant passage  $d^-$ .

### 3.2 Query Likelihood Modeling (QLM)

The objective of document ranking revolves around estimating  $P(d|q)$ , which represents the likelihood that a document  $d$  is pertinent to a given query  $q$ . Leveraging Bayes’ Theorem, we can express this probability as follows:

$$P(d|q) = P(q|d)P(d)/P(q), \quad (2)$$

where  $P(q)$  is the probability of the query  $q$ .  $P(q)$  is the same for all documents, and so can be ignored. The prior probability of a document  $P(d)$  is often treated as uniform across all  $d$  and so it can also be ignored. Therefore, QLM return results are ordered primarily based on  $P(q|d)$ .

**Query Likelihood.** Given a query  $q = \{q_1, q_2, \dots, q_n\}$ , where  $q_i$  is the  $i$ -th term of the query  $q$ , compute the likelihood of document  $d$  generating the query  $q$ . Assuming query terms are generated independently, then the query likelihood is computed:

$$P(q|d) = \prod_{i=1}^n P(q_i|d), \quad (3)$$

In order to compute  $P(q_i|d)$ , there are two methods, i.e., (1) Statistical language models, which estimate the term frequency as the  $P(q_i|d)$ . However, the term frequency of those terms that do not appear in the document will be zero. Therefore, smoothing methods have been commonly used, e.g., Dirichlet Smoothing and Jelinek-Mercer Smoothing [66]. (2) Neural language models that directly predict the query term generated probability based on the input document. Specifically, the probability of sampling the next query term is conditioned on the document and all previous query terms, i.e.,

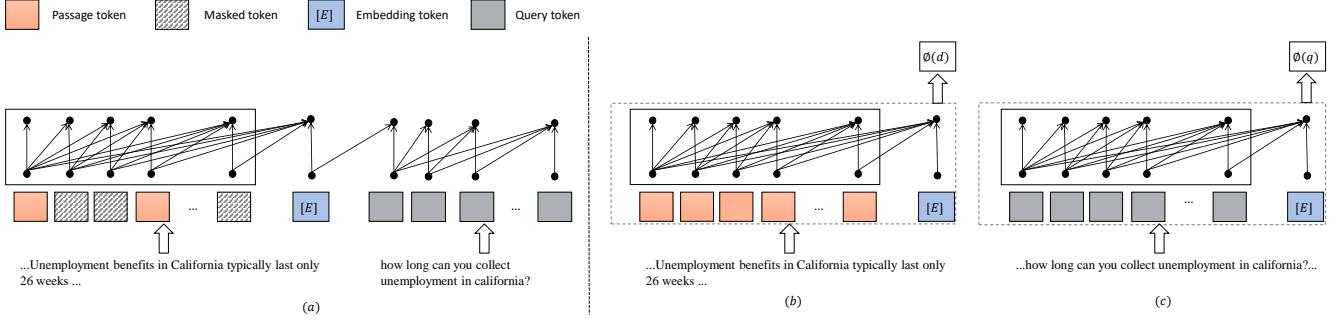
$$P(q_i) = P(q_i|d, q_1, q_2, \dots, q_{i-1}). \quad (4)$$

Therefore, the query likelihood is:

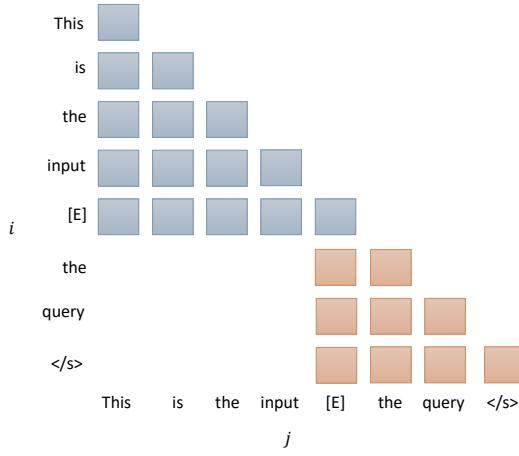
$$P(q|d) = \prod_{i=1}^{|q|} P(q_i). \quad (5)$$

## 4 Method

As we all know, the generative capabilities of decoder-only architecture large language models are highly potent. Although unidirectional attention can help LLMs have a natural advantage in general pre-training, it may have the defect of focusing more on local representations in dense retrieval [4, 33]. Therefore, there are many studies have devised many strategies to enhance the embedding of LLMs’ global semantic representation. Unlike these works, we focus on leveraging the generation capabilities of LLMs and unleashing their potential in document representations for retrieval. Query likelihood modeling estimates the probability that a document generates a query as a representation of the document. Therefore, we propose LLM-QL, which introduces query likelihood modeling in dense retrieval and aims to adapt LLMs’ generation capabilities for dense retrieval. In order to condense the semantics of documents or passages to a single vector, we propose two strategies in LLM-QL: **Attention Stop (AS)** and **Input Corruption (IC)**. Overall, LLM-QL contains two-stage training: query likelihood learning (QL learning) and contrastive fine-tuning, as detailed in Figure 1. After experiments, we found that the performance improvement is very remarkable compared to other baselines. Next, we will show the details of our LLM-QL.



**Figure 1: The details of LLM-QL.** a) QL learning: The input passage is moderately masked (the orange rectangles and slash rectangles). When generating a query (the gray rectangles), only the [E] part (the blue rectangle) can be seen at the farthest. b) and c) Contrastive learning: The LLMs encode up to the [E] position at the farthest. The representations at the [E] position are used to represent the embeddings of the query and the passage, respectively. In our experiments, we use the special token </s> as the [E].



**Figure 2: The attention scheme of LLM-QL.**

#### 4.1 QL learning

QL learning contains two strategies, i.e., **Attention Stop (AS)** and **Input Corruption (IC)**.

**Attention Stop (AS).** AS means that during query generation learning, we stop the attention at the sentence ending token, i.e., </s>, so that the passage semantics are forced to be compressed to this single token for predicting the query. Figure 2 shows the attention stop scheme. In the self-attention mechanism, given an input sequence represented as  $(X \in \mathbb{R}^{T \times d})$ , where  $T$  is the sequence length and  $d$  is the feature dimension. Queries ( $Q$ ), Keys ( $K$ ), and Values ( $V$ ) are obtained through linear transformations:

$$Q = XW^Q, K = XW^K, V = XW^V, \quad (6)$$

where  $(W^Q, W^K, W^V \in \mathbb{R}^{d \times d_k})$  are learnable parameter matrices. Compute the original attention weights:

$$A_o = \text{softmax}\left(\frac{QK^T}{d_k}\right). \quad (7)$$

We introduce a mask matrix  $(M \in \mathbb{R}^{T \times T})$  for self-attention mechanism to achieve attention stop:

$$M_{ij} = \begin{cases} 0, & \text{if } i \leq I_e, j \leq I_e, j \leq i, \\ 0, & \text{if } i > I_e, j \geq I_e, j \leq i, \\ -\infty, & \text{otherwise,} \end{cases} \quad (8)$$

$M_{ij}$  means whether the  $i$ -th token can pay attention to the  $j$ -th token or not. This mask ensures that when computing the attention for position  $i$  ( $i \leq I_e$ ,  $I_e$  is the position of [E]), only the information from positions up to  $i$  is considered, and attention for position  $i$  ( $i > I_e$ ), only the information from positions up from  $I_e$  to  $i-1$  is considered. Positions, which are masked by  $-\infty$ , effectively set their attention weights to zero after applying the softmax function. Combining the  $M$  and  $A_o$ , the computation process for final attention is:

$$A = \text{softmax}\left(\frac{QK^T}{d_k} + M\right). \quad (9)$$

With the mask attention, [E]'s role is to represent the  $d$  and predict the corresponding query  $q$ .

**Input Corruption (IC).** Inspired by the success of corruption in document representation [8], we randomly mask the input passage tokens. Give the input  $X = \{I_1, d_1, \dots, d_n, I_2, [E], q_1, \dots\}$ , where  $I_1$  is the prefix prompt “Instruct: Given a retrieved passage, summarize the passage. Passage:”,  $I_2$  is the post prompt “Summarization:”,  $d_i$  is the token of the passage with a probability of  $p\%$  to be masked, and  $q_i$  is the token of the corresponding query. Following BehnamGhader et al. [4], we use the token “\_” as the masked token. Then the masked input is  $X' = \{I_1, d_1, \dots, \_, \dots, d_n, I_2, [E], q_1, \dots\}$ .

The QL learning is training on q-d pairs from the MS MARCO training set and optimized by maximizing query likelihood:

$$\mathcal{L}_q = - \sum_i^n \log(p(q_i | I_1, d_1, \dots, \_, \dots, d_n, I_2, [E], q_1, \dots, q_{i-1})). \quad (10)$$

#### 4.2 Contrastive Learning

Following previous fine-tuning work, RepLLaMA [39], we also use the end-of-sequence token  $E$  to the input query or passage to form the input sequence to LLMs. Take LLaMa as an example, the dense

embedding of a query  $\phi(q)$  is computed as:

$$\phi(q) = \text{LlaMa}(I_1^q, q, I_2^q, E)[E], \quad (11)$$

where  $I_1^q, I_2^q$  are “Instruct: Given a web search query, retrieve the most relevant passage that answers the query. Query:” and “The most relevant passage:”, respectively. Note that adding additional query2doc to QL learning does not bring a gain in retrieval performance, refer to section 7.1.2 for details.

The dense embedding of a passage is computed as:

$$\phi(d) = \text{LlaMa}(I_1^d, d, I_2^d, E)[E], \quad (12)$$

where  $I_1^d, I_2^d$  are “Instruct: Given a retrieved passage, summarize the passage. Passage:” and “Summarization:”, respectively.

Relevance of  $d$  to  $q$  is computed in terms of the dot product of their corresponding dense representation  $\phi(q)$  and  $\phi(d)$  as :

$$\text{Sim}(q, d) = \langle \phi(q), \phi(d) \rangle, \quad (13)$$

The LLM is then optimized end-to-end according to InfoNCE loss:

$$\mathcal{L}_c(q, d^+, D^-) = -\log \frac{\exp(\text{Sim}(q, d^+))}{\sum_{d \in \{d^+ + D^-\}} (\text{Sim}(q, d))}, \quad (14)$$

where  $d^+$  represents a passage that is relevant to the query  $q$ , while  $D^-$  denotes a set of passages that is not relevant to the query. The set of negative passages  $D^-$  includes both hard negatives, which are sampled from the top-ranking results of an existing retrieval system (i.e., BM25 and CoCondenser [16]), and in-batch negatives, which are derived from the positive passages and hard negative passages associated with other queries in the same training batch.

**Discussion.** About how to incorporate the QL loss, we also tried to combine the query likelihood loss  $\mathcal{L}_q$  and the contrastive learning loss  $\mathcal{L}_c$ , but found it is better than using  $\mathcal{L}_c$  alone while worse than incorporating query generation learning as an additional training phase. It is challenging to find the optimal effect  $\mathcal{L}_q$  should take when combined with  $\mathcal{L}_c$ . So, we omit the choice of a weighted combination of the two. About large-scale pretraining with LLM-QL, if we collect pseudo queries for more documents, e.g., by predicting with a generative model like an LLM, LLM-QL can be adapted to large-scale pretraining. This is a viable pretraining-for-IR approach and may achieve promising results. However, this is beyond the scope of this paper, and we leave it for future work.

### 4.3 Inference

The retrieval corpus contains passages at the million-scale or even higher. To achieve millisecond-level latency for querying the corpus, dense retrieval indexes the entire corpus offline using  $\phi(d)$  function and then conducts searches using the approximate nearest neighbor (ANN) [26] search method, identifying the top- $k$  closest document vectors to the query embedding  $\phi(q)$  using the dot product function  $\text{Sim}(q, d)$  in Equation 13.

## 5 Experimental Setup

### 5.1 Datasets

We use the MS MARCO [3] passage retrieval dataset in our experiments. It consists of around 8.8 million passages. Following, we train our model on MS MARCO training data, containing 502,939 queries. We evaluate all the models on MS MARCO-dev, TREC-DL19, and TREC-DL20 [10] datasets. MS MARCO-dev includes 6,980 queries,

which have an average of 1.1 relevant passages per query. Following standard practice, we adopt MRR@10 and recall@1000 as the main evaluation metrics on MS MARCO-DEV in our experiments. TREC DL 19 and 20 contain 43 and 54 judged queries, respectively. Each of the queries of TREC-DL 19 and 20 has 95.4 and 66.8 relevant passages, respectively. Judgments of TREC DL are on a four-point scale, i.e., “perfectly relevant”, “highly relevant”, “related”, and “irrelevant”. Therefore, such data can be used to evaluate fine-grained ranking performance. We report the performance using NDCG@10 on the TREC DL in our experiments. Detailed statistics of all the experimental datasets are shown in Table 1.

	MS MARCO-dev	TREC DL-19	TREC DL-20
#Queries	6980	43	54
#Rel.Passage	7437	4102	3606
#Rel.Passage/q	1.1	95.4	66.8
#Graded.Labels	2	4	4

**Table 1: Statistics of MS MARCO-dev, TREC DL 19, and TREC DL 20.**

Furthermore, we assess the zero-shot performance of our LLM-QL on the BEIR benchmark [52], which encompasses a diverse array of 18 datasets from various fields (e.g., news and medicine), and contains a range of retrieval tasks (e.g., fact-checking and question answering). We chose some tasks that are publicly available in BEIR to evaluate the models, i.e., T-COVID [55], NFCorpus [6], NQ [30], HotpotQA [65], FiQA [40], ArguAna [56], Touche [5], Quora [52], DBpedia [18], SCIDOCs [9], FEVER [53], C-FEVER [12], SciFact [57], and CQA [20].

### 5.2 Training Settings

LLM-QL is applied to the LLaMA-2-7B (base) model. We design two training tasks for LLM-QL, i.e., query likelihood learning and contrastive learning fine-tuning. LLM-QL is trained with Deepseed [48] which is an efficient deep-learning optimization library, and Zero Redundancy Optimizer-3 (ZeRO-3), which is a family of memory optimization technologies for large-scale distributed deep learning.

**QL learning.** For the QL learning, we perform 2 epochs on MS MARCO q-d pairs in total, with a batch size of 512, query max length of 200, passage max length of 200, and a learning rate of 1e-5. The default masking ratio of the passage is 0.6. The training is on a machine with 8× Nvidia A800 (80GB) GPUs. We use full parameter fine-tuning during QL learning.

**Contrastive Learning.** For the second stage training, following RepLLaMA [39], LLM-QL also leverages LoRA [21] for the parameter-efficient training of LLMs, and simply relies on the hard negatives to fine-tune our LLM-QL with contrastive learning. In line with RepLLaMA, we employ a combination of BM25 and CoCondenser [16] to generate hard negatives, ensuring that these hard negative examples are sourced from both sparse and dense retrieval outcomes. Following the training details of our relevant baseline [39], the model is trained for 1 epoch, batch-size 32, learning rate 1e-4, and gradient accumulation steps 4 on 8× Nvidia A800 (80GB) GPUs.

**Table 2: MS MARCO-dev passage retrieval (performance measured by MRR@10, Recall@1000, NDCG@10).** \* means that we reproduce the models using LLaMa2 according to the original paper. ‘-’ indicates that the original paper does not specifically mention the corresponding terms. ‘†’ indicate significant improvements over Summarize, Echo, LLM2VEC ( $p < 0.05$ ) and Hommel’s correction ( $\alpha = 0.05$ ).

-	-	-	-	-	Dev	DL’19	DL’20
Method	Pre-training	hard negatives	Size	FT.	MRR@10	Recall@1000	NDCG@10
BM25 [34]	No	-	-	-	18.4	85.3	50.6
DPR [28]	No	Static(BM25)	110M	hard	31.4	95.3	59.0
ANCE [62]	No	Dynamic	125M	hard	33.0	95.9	64.8
ADORE [67]	No	Dynamic	110M	hard	34.7	-	68.3
Condenser(BM25) [15]	Yes	Static(BM25)	110M	hard	33.8	96.1	64.8
coCondenser [16]	Yes	Dynamic	110M	hard	38.2	98.4	71.2
SimLM [58]	Yes	Dynamic	110M	hard	39.1	98.6	69.8
RetroMAE(BM25) [60]	Yes	Static(BM25)	110M	hard	35.5	97.6	-
TAS-B [19]	No	-	55M	distill	34.3	97.6	72.2
SimLM+distill [58]	Yes	Dynamic	110M	distill	41.1	98.7	71.4
RocketQAv2 [49]	No	Dynamic	-	distill	38.8	98.1	-
RetroMAE+distill [60]	Yes	Dynamic	110M	distill	41.6	98.8	68.1
DRAGON [35]	Yes	Dynamic	110M	distill	39.0	98.6	74.4
I3retriever4 [13]	Yes	Dynamic	-	distill	41.8	98.8	73.1
GTR-XXL [44]	Yes	Static(RocketQA)	4.8B	-	38.8	99.0	-
OpenAI-Ada-002 [43]	Yes	-	-	-	34.4	98.6	70.4
RepLLaMA (LoRA) [39]	No	Static(CoCondenser+BM25)	7B	hard	41.2	99.4	74.3
RepLLaMA (FT) [39]	No	Static(CoCondenser+BM25)	7B	hard	41.6	-	72.8
Summarize* (LoRA) [25]	No	Static(CoCondenser+BM25)	7B	hard	41.0	99.4	73.1
Echo* (LoRA) [51]	No	Static(CoCondenser+BM25)	7B	hard	41.5	99.4	74.5
LLM2VEC* (LoRA+LoRA) [4]	No	Static(CoCondenser+BM25)	7B	hard	-	-	-
LLM2VEC* (FT+LoRA) [4]	No	Static(CoCondenser+BM25)	7B	hard	41.9	99.3	72.9
LLM-QL (LoRA+LoRA)	No	Static(CoCondenser+BM25)	7B	hard	41.9	99.4	73.8
LLM-QL (FT+LoRA)	No	Static(CoCondenser+BM25)	7B	hard	42.4 †	99.4	73.6

### 5.3 Baselines

We make comparisons with a wide variety of baseline methods on dense retrieval, i.e., (1) **Basic lexical retriever**: BM25 [34]. (2) **PLM-s-based dense retrievers**: DPR [28], ANCE [62], ADORE [67], Condenser [15], RocketQAv2 [49], coCondenser [16], RetroMAE [60], SimLM [58], TAS-B [19], I3Retriever [13], and DRAGON [35]. (3) **LLMs-based dense retrievers**: GTR-XXL based on T5-4.8B [44], SGPT [42], OpenAI-Ada-002 based on GPT [43], RepLLaMA [39] based LLaMa 2 7B [54], Echo embedding [51], Summarize embedding [25], LLM2VEC [4] based on LLaMa 2 7B. LLM-based retrievers are the closest baselines to our method.

- **RepLLaMa:** Extracting the final layer hidden state representation of the `</s>` token as the dense representation for the query or the passage, i.e.,

$$\phi(q) = \text{LlaMa}(\text{"Query : ", } q)[-1], \quad (15)$$

- **Echo embedding (Echo):** During fine-tuning the LLMs, repeat the input twice in context and extract embeddings from the second occurrence, i.e.,

$$\phi(T) = \text{LlaMa}(I_1, T, I_2, T')[-1], \quad (16)$$

where  $I_1$  is ‘Rewrite the following sentence:’ and  $I_2$  is ‘The rewritten sentence:’.

- **Summarize embedding (Summarize):** During fine-tuning the LLMs, summarize the input text into one word, i.e.,

$$\phi(T) = \text{LlaMa}(I_1, T, I_2)[-1], \quad (17)$$

where  $I_1$  is ‘This sentence:’ and  $I_2$  is ‘means in one word:’.

- **LLM2VEC:** LLM2VEC speculates that the limited use of decoder-only LLMs in text embedding is partly due to their causal attention approach, which restricts the creation of rich contextual representations. So LLM2VEC directly replaces the causal attention mask of decoder-only LLMs with an all-one matrix. However, simply enabling bidirectional attention does indeed decrease embedding performance for LLMs. Therefore, LLM2VEC introduces the masked next token prediction task to make the LLMs aware of its bidirectional attention, i.e., the mask matrix ( $\mathbf{M} \in \mathbb{R}^{T \times T}$ ) in Equation 8 is  $M_{ij} = 0$ .

For a fair comparison, we uniformly selected LLaMa 2 7B [54] as our embedding model and took the last token pooling method during fine-tuning to get the embedding of the query or passage. For the original baselines that did not use LLaMa 2 7B [54], we reproduced them in LLaMa 2 7B according to the method of the original paper. Both LLM2VEC and our model are two-stage training. For the first-stage training, we used full-parameter fine-tuning for the first-stage training and LoRA fine-tuning for the second stage.

## 6 Experimental Results

### 6.1 Supervised Performance

The experiment results on MS MARCO passage retrieval and TREC-DL datasets are shown in Table 2. We make a very extensive comparison, which contains pre-training, hard negative sampling, and

**Table 3: Zero-shot retrieval on BEIR benchmark. (The performance is measured by NDCG@10). For the first-stage training of two-stage training retrievers, we used full-parameter fine-tuning (FT) and LoRA fine-tuning (LA) strategies, respectively.**

method size	BM25 110M	BRET 4.8B	GTR-XXL 175B	CPT-XL 5.8B	Ada-2 7B	SGPT 7B	Repllama 7B	LLM2VEC(FT) 7B	LLM2VEC(LA) 7B	LLM-QL (FT) 7B	LLM-QL (LA) 7B
<b>BEIR Search Tasks</b>											
DBpedia	31.8	31.4	40.8	43.2	40.2	39.9	43.7	45.1	<u>45.1</u>	44.0	<b>45.2</b>
FiQA	23.6	25.2	<u>46.7</u>	<b>51.2</b>	41.1	37.2	45.8	44.8	39.6	44.7	46.2
NQ	30.6	46.7	56.8	-	48.2	52.4	62.4	62.4	62.8	63.1	<b>64.9</b>
HotpotQA	63.3	48.8	59.9	68.8	65.4	59.3	68.5	69.1	69.3	67.8	<b>69.4</b>
NFCorpus	32.2	26.0	34.2	<b>40.7</b>	35.8	36.2	37.8	36.2	33.7	36.6	<u>36.8</u>
T-COVID	59.5	61.5	50.1	64.9	81.3	<b>87.3</b>	84.7	<u>84.8</u>	80.8	83.7	84.0
Touche	<b>44.2</b>	25.9	25.6	29.1	28.0	25.4	30.5	39.1	32.3	32.1	<u>37.0</u>
CQA	32.5	28.2	39.9	-	41.7	38.1	37.4	41.3	37.0	39.4	<b>41.8</b>
Avg	39.7	36.7	44.3		47.7	47.0	51.4	<u>52.8</u>	50.1	51.4	<b>53.2</b>
<b>BEIR Semantic Relatedness Task</b>											
ArguAna	39.7	26.5	<b>54.0</b>	43.5	<b>56.7</b>	51.4	48.6	40.8	40.5	40.2	
C-FEVER	16.5	18.7	26.7	22.3	23.7	30.5	<b>31.0</b>	30.3	25.3	24.9	<u>30.4</u>
FEVER	65.1	68.2	74.0	77.5	77.3	78.3	<b>83.4</b>	78.0	70.0	76.8	<u>82.8</u>
Quora	78.9	78.7	<b>89.2</b>	63.8	87.6	84.6	86.8	87.7	88.4	<u>88.6</u>	87.6
SCIDOCs	14.1	11.3	16.1	-	18.6	<b>19.7</b>	18.1	17.7	17.6	18.9	<u>19.0</u>
SciFact	67.9	53.3	66.2	75.4	73.6	74.7	<u>75.6</u>	<b>76.6</b>	73.2	75.5	73.6
Avg	47.0	42.8	54.4	-	56.3	<u>56.5</u>	<b>57.3</b>	55.2	52.5	54.2	55.6
Total Avg	42.9	39.3	48.6	-	51.4	51.1	<u>53.9</u>	53.8	51.1	52.6	<b>54.2</b>

fine-tuning (i.e., FT, in 2). The pre-training method refers to the unsupervised pre-training task in a large corpus, e.g., Wiki, MS MARCO corpus. Hard negatives play a very important role in dense retrieval training methods. There are two kinds of hard negatives, namely static and dynamic ones [67]. The static ones adopt a traditional retriever, e.g., BM25 [34], or a warm-up dense retrieval model, e.g., CoCondenser [16], to retrieve the top results as unchanging hard negatives during training. The dynamic ones rely on the current retriever and retrieve the top results using its previous checkpoint as the hard negatives in the next step of training. For fine-tuning methods, there are two different types: one is based on hard-negative sampling (hard); the other one is based on knowledge distillation (distill) from a cross-encoder ranker. According to past experience, a dense retriever trained with pre-training, dynamic hard negative sampling, and knowledge distillation will have better performance compared to those without the pre-training method, static hard negative sampling, or knowledge distillation, respectively, but have a larger cost.

**PLMs-based Retrievers vs LLMs-based Retrievers.** From Table 2, we can observe that LLMs-based retrievers generally have better performance than PLMs-based retrievers. For example, RepLLaMA has 6% improvement in terms of MRR@10 on MS MARCO-dev compared to RocketQAv2. The best PLMs-based retrievers, which contain a pre-training stage, dynamic hard negatives, and a knowledge distillation fine-tuning method, are comparable to the most basic LLMs-based retrievers. This indicates the superior performance of LLMs in dense retrieval.

**Uni-directional Attention vs Bi-directional Attention.** Many studies suggest that bidirectional attention is more beneficial for capturing global semantic information compared to unidirectional

attention. Without any additional operations, bidirectional attention outperforms unidirectional attention in tasks involving retrieval, as seen in comparisons like LLM2VEC vs RepLLaMA in Table 2. However, this does not imply that unidirectional attention is unsuitable for retrieval tasks. By incorporating extra training methods into the decoder-only LLMs, e.g., simple generation training, unidirectional attention can also achieve superior retrieval performance to bidirectional attention.

**LLM-QL vs Others.** LLM-QL achieves a superior retrieval performance in every evaluation dataset. We can achieve performance comparable to other fine-tuned LLM-based retrievers. Table 7 shows the performance comparison on different metrics. Remarkably, LLM-QL achieves **MRR@10: 42.4** and **NDCG@20: 51.7** on MS MARCO-dev, having significant performance improvement compared to baselines. LLM-QL improves the performance of the baselines and presents a new state-of-the-art result on the large-scale dev set. Its performance is slightly lower on DL'19 and DL'20, due to the randomness of the small test set.

## 6.2 Zero-shot Performance

We delve deeper into LLM-QL's influence on generalization capabilities by leveraging the BEIR benchmark alongside the LLM-QL for evaluating its zero-shot performance. Both LLM2VEC and our LLM-QL are two-stage training. LLM2VEC contains bi-attention with mask next token prediction (Bi+MNTP) and contrastive Learning. For the first-stage training, we used full-parameter fine-tuning (FT) and LoRA fine-tuning strategies (LA), respectively. Moreover, for the supervised performance of FT and LA on the first stage training, LLM-QL is 42.4 vs 41.9 on MRR@10; LLM2VEC is 41.9 vs 41.7 on MRR@10, indicating that full-parameter fine-tuning in the first stage has better performance on supervised performance. We

**Table 4: Ablation studies on AS, IC of QL, and skip QL on the MS MARCO-dev. “R@k” means “Recall@k”.**

Corruption ratio	MRR@1000	NDCG@10	R@10	R@1000
0.0 (w. AS)	38.8	44.5	67.6	99.1
0.2 (w. AS)	41.6	47.5	70.4	99.4
0.4 (w. AS)	42.6	48.3	71.1	99.4
0.6 (w. AS, LLM-QL)	<b>43.5</b>	<b>49.2</b>	<b>71.8</b>	<b>99.4</b>
0.8 (w. AS)	43.1	48.9	71.7	99.4
0.6 (w/o AS)	43.2	49.0	71.8	99.4
Skip QL	42.7	48.5	71.5	99.3

divided the 14 datasets into two task types: search and semantic relatedness tasks, as shown in Table 3. We can observe that (1) LLMs-based retrievers outperform BM25 and BERT-based retrievers on BEIR, demonstrating the powerful generalization capabilities inherent in large language models. (2) LLM-QL exhibits a remarkable performance compared to other methods on BEIR, especially on search tasks, where it achieves an average performance of 54.2. (3) Although LLM2Vec and our LLM-QL outperform Repllama on the Dev test set, their performance on the BEIR Zero-shot test is inferior to that of Repllama’s semantic track. The reason might be that the training is oriented towards search tasks, and the additional strategies for performance improvement proposed are primarily tailored for the search track.

## 7 Further Analyses

In this section, we conduct a thorough analysis of LLM-QL to clarify its advantages.

### 7.1 Ablation studies

To better compare the performance differences between models, we use MRR@1000 and NDCG@10, Recall@10, and Recall@1000 for evaluation.

**7.1.1 Impact of Attention Stop (AS) and Input Corruption (IC).** Table 4 shows the different performance of LLM-QL with different corruption ratios for IC, performance without AS, and skip QL performance.

**Corruption Ratio.** A small corruption ratio may not be sufficient to assist LLMs in better condensing the semantics of passages. Conversely, an excessively large corruption ratio can hinder the model from comprehending the input passage and compress it into a query, as it becomes challenging for the model to process.

**AS vs IC.** From Table 4, we found that removing AS or IC both has an impact on the performance of the model. However, the importance of the two for performance gains is not the same. For example, removing IC (i.e., corruption ratio is 0) and AS (i.e., 0.6(w/o MA) in Table 4) reduces the performance on the MRR@1000 metric by 10.7% and 0.7%, respectively. Removing IC has a greater impact on performance than removing AS, which illustrates the importance of the corruption mechanism.

**QL vs Skip QL.** We used the same input instruction for the query and documents during fine-tuning without QL learning, and the performance declined, indicating the importance of QL learning.

**7.1.2 Query Expansion and Doc2query.** In the contrastive Learning stage, the instructions on the query and passage sides are different.

**Table 5: Different performance on MS MARCO-dev using different first-stage generation training methods.**

Training	MRR@1000	NDCG@10	Recall@10	Recall@1000
Doc2query (LLM-QL)	<b>43.5</b>	<b>49.2</b>	<b>71.8</b>	<b>99.4</b>
Query2doc	39.3	45.0	67.9	99.3
Hybrid	40.9	46.7	69.4	99.3

**Table 6: Performance of retrieval and re-ranking task using query likelihood model.**

Method	NDCG@1	NDCG@3	NDCG@10	MRR@10
BM25	10.54	16.96	22.84	18.40
QLM-JM	9.60	15.86	21.81	17.40
QLM-D	8.31	13.71	18.74	14.91
BM25 + LLaMa 2	0.95	1.66	2.94	2.12
BM25 + LLM-QL				
+ (w/ AS, w/o FT)	15.99	26.03	33.67	27.58
+ (w/o AS, w/o FT)	17.08	27.10	34.75	28.65
BM25 + LLM-QL	<b>21.62</b>	<b>32.23</b>	<b>39.04</b>	<b>33.30</b>

**Table 7: MS MARCO passage retrieval performance.**  $\dagger$  means indicates significant improvements over all baselines ( $p < 0.05$ ) and Hommel’s correction ( $\alpha = 0.05$ ). “M@k”, “N@k” means “MRR@k”, “NDCG@k”, respectively.

Model	M@10	M@20	M@100	N@10	N@20	N@100
Echo	41.49	42.14	42.49	48.60	51.02	53.54
Summarize	40.97	41.65	42.00	47.99	50.52	53.06
LLM2VEC	41.86	42.49	42.85	48.83	51.19	53.75
LLM-QL	42.41 <sup>†</sup>	43.08 <sup>†</sup>	43.43 <sup>†</sup>	49.17	51.69 <sup>†</sup>	54.21 <sup>†</sup>

We only used the instructions on the passage side in the QL learning process. We employed query-side instructions (query2doc) and both instructions (hybrid) in the QL process. The results are shown in Table 5. The hybrid retrieval performance is between query2doc and doc2query. Query2doc has the worst retrieval performance on all scores, while doc2query has the best retrieval performance. We can infer that QL learning does not require additional query instruction training, which also illustrates the importance of passage compression in the dense retrieval stage and verifies the effectiveness of our method. Prior research [13, 45] also demonstrated the significance of doc2query compression in dense retrieval.

## 8 Impact of [E]

Regarding the choice of [E], we considered and analyzed three types from the perspective of quantity and type: (1) </s>, (2) </s><sup>\*4</sup>, and (3) <s1><s2><s3><s4>. For multiple vectors, we take the average of these vectors as the representation of the query or passage during fine-tuning. The performance of the three results after fine-tuning is shown in Table 9. We can observe that (1) Using the LLMs’ own special token has better retrieval performance compared to a special token like <s1>. (2) For MRR and NDCG scores, using a single vector </s> has better performance than multiple vectors. For recall scores, using multiple vectors to represent performance is better than using a single vector.

**Table 8:** The cases of doc2query generated by our LLM-QL trained with the first stage. The red texts represent those terms that are consistent with the topic of the relevant query. The blue texts represent those terms that are inconsistent with the topic of the relevant query, but the terms appear in the passage.

Passage	McDonald's Corporation is one of the most recognizable corporations in the world. A <b>corporation</b> is a company or group of people authorized to act as a single entity (legally a person) and recognized as such in law...
Relevant query	what is a <b>corporation</b> ?
Generated queries (Top-5)	1 . what is a <b>corporation</b> 2. what is a <b>corporation</b> ? 3. what is <b>corporation</b> 4. what is corporate 5.what is <b>mcdonalds</b>
Passage	The <b>symptoms</b> are similar, but the <b>mouse</b> will be in much worse condition: runny eyes; sneezing.... To prevent influenza, do not touch your pet if you have <b>flu</b> , as mice catch it from humans.
Relevant query	<b>symptoms</b> of a dying <b>mouse</b>
Generated queries (Top-5)	1. sympt ill' 2. sympt <b>symptoms</b> . 3.sympt <b>symptoms</b> 4.what are the <b>symptoms</b> of the <b>flu</b> 5.what are the <b>symptoms</b> of a sick <b>mouse</b>

**Table 9:** Ablation studies on [E] on MS MARCO-dev. “</s>\*4” is “</s></s></s></s>”. “s1234” is <s1><s2><s3><s4>.

[E]	MRR@1000	NDCG@10	Recall@10	Recall@1000
</s> (LLM-QL)	<b>43.5</b>	<b>49.2</b>	71.8	99.4
</s>*4	43.1	49.0	<b>72.1</b>	<b>99.5</b>
s1234	43.1	48.9	71.8	99.4

## 8.1 Query Likelihood Modeling Performance

We conducted an experimental analysis on the traditional QLM, as shown in Table 6. For retrieval, we choose the BM25( $k_1 = 0.9$  and  $b = 0.4$ ), which is utilized with Anserini with default parameters [63], and some traditional QLM methods, i.e., (1) Query Language Models with Dirichlet smoothing (QLM-D), (2) Query Language Models with Jelinek-Mercer smoothing (QLM-JM). We can observe that traditional QLM methods have poor performance compared to BM25. To verify the performance of the generated queries after the first-stage generation training of LLM-QL, we substitute the LLM-QL for the MLE (Maximum Likelihood Estimation) in QLM (Query Likelihood Modeling). Considering the high cost of LLMs in retrieval using the QLM method, we only apply the ranking task using LLM-based QLM. We use the BM25 on the MS MARCO passage top-200 results as the first-stage retrieval results and then use LlaMa 2 and LLM-QL to carry out the passage ranking task, respectively. Table 6 shows the performance of reranking on these two models. We can observe that: (1) The LlaMa 2 without special training has almost no ability to use query likelihood for ranking. This shows that although the LLMs have good generative ability after large-scale pre-training, they still need some special training to activate the generative ability for specific tasks. (2) Using LLM-QL to perform reranking tasks based on query likelihood is much better than LlaMa 2. This indicates that LLM-QL can generate high-quality queries. (3) Through contrastive learning training, the performance of the reranking has been further improved. The reason may be that the input corruption in the first stage, which implemented extensive token-level dropout on the text inputs, may could potentially impair the generative capabilities of the LLM.

**Table 10:** “FT Batch-size”, “TIT”, and “TTT” means Batch-size during fine-tuning, Total Index Time, and Total Training Time, respectively.

Method	FT Batch-size	Index Size	TIT	TTT	MRR@10
RetroMAE	128	26G	13m	12h	35.5
RepLLaMa	32	136G	3h	17h	41.2
Echo	32	136G	9h	37h	41.5
LLM2VEV	32	136G	4h	24h	41.9
LLM-QL	32	136G	4h	23h	42.4

## 8.2 More Comparison on Large-scale dataset

Table 2 mainly shows MRR@10 and Recall@1000, and Recall@1000 has already reached 99.4. Several LLM-based retrievers are relatively close on the Recall@1000 score and cannot be compared. To explain the advantages of our LLM-QL more clearly, we compared other indicators between our LLM-QL and three baselines with superior performance, i.e., Echo, Summarize, and LLM2VEC. The experimental results are shown in Figure 7. Through the results, we found that our LLM-QL passed the significance test ( $p < 0.05$ ) on all indicators, indicating that our model is significantly better than other baselines on a large-scale test set.

## 9 Case Study

To better understand the performance of query likelihood generation training, we demonstrate two cases in Table 8, and interpret their query reconstruction. Specifically, we use the first-stage QL learning trained model to explicitly generate doc2query for two q-d pairs in MS MARCO-dev using beam search to get the top-5 generated queries. The attention distribution of the generation process is shown in Equation 8. We can see that (1) The key terms in the top-k queries generated by the two passages are basically consistent with the corresponding queries, which shows the effectiveness of our LLM-QL training. The model can generate very relevant query expressions based on the representation at [E]. (2) In the queries generated by the second, more complex example, some queries’ terms do not match the key term of the corresponding relevant query. This shows that it is difficult to perfectly generate corresponding queries based on [E] for complex passages compared to simple passages.

## 10 Efficiency

Table 10 shows the efficiency comparison of LLM-QL, the PLMs-based model, and the other three LLMs-based models. For LLMs for dense retrieval, training cost as well as inference cost need to be considered. We report the Index size, all training time, and index time of the total corpus as the key metrics.

**Total Training Time.** We test all the models on 8xA800 80G for fair comparison and show the results in Table 10. For LLM-based retrievers, it demonstrates a large cost compared with ANCE. The echo method has the largest cost but only has a little gain compared to RepLLaMA. The primary reason for the high training cost of large models stems from their extensive model parameters, leading to prolonged forward propagation times. However, it is intriguing to note that despite RepLlama having 63 times more parameters than RetroMAE, its running cost is only 1.4 times higher.

**Inference time.** Due to the fact that the LLMs' representation dimension is 4096, whereas RetroMAE's representation dimension is 768, the resulting index space occupancy is relatively large, which can impact the efficiency of inference. As pointed out in previous work [33], altering the representation dimension of large models can lead to a reduction in their retrieval performance. This highlights the need for continued research in this area in the future.

## 11 Conclusion

In this work, we propose LLM-QL inspired by the query likelihood modeling, which aims to utilize generation capabilities for dense retrieval. LLM-QL contains two training stages: QL learning and Contrastive Learning. For QL learning, in order to sufficiently condense the semantics of documents or passages to a single vector, we propose two strategies **task contains Attention Stop (AS)** and **Input Corruption (IC)**. Specifically, AS means that during query generation modeling, we stop the attention at the sentence ending token, i.e., </s>, so that the document semantics are forced to be compressed to this single token for predicting the query. IC means corrupting the document by randomly masking a portion of tokens, aiming to condense as many document semantics as possible to the final representation and improve training efficiency. Through this simple step of adaptation, our method remarkably enhances the model's fine-tuned performance on dense retrieval benchmarks. We also conduct comprehensive analyses of the components of LLMs, other options of the model, and the ranking performance of query likelihood estimation based on LLM-QL.

## References

- [1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).
- [2] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609* (2023).
- [3] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268* (2016).
- [4] Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. 2024. Llm2vec: Large language models are secretly powerful text encoders. *arXiv preprint arXiv:2404.05961* (2024).
- [5] Alexander Bondarenko, Maik Fröbe, Meriem Beloucif, Lukas Giennapp, Yamen Ajjour, Alexander Panchenko, Chris Biemann, Benno Stein, Henning Wachsmuth, Martin Potthast, et al. 2020. Overview of Touche 2020: argument retrieval. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction: 11th International Conference of the CLEF Association, CLEF 2020, Thessaloniki, Greece, September 22–25, 2020, Proceedings 11*. Springer, 384–395.
- [6] Vera Boteva, Demian Gholipour, Artem Sokolov, and Stefan Riezler. 2016. A full-text learning to rank dataset for medical information retrieval. In *Advances in Information Retrieval: 38th European Conference on IR Research, ECIR 2016, Padua, Italy, March 20–23, 2016. Proceedings 38*. Springer, 716–722.
- [7] Wei-Cheng Chang, Felix X Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. 2020. Pre-training tasks for embedding-based large-scale retrieval. *arXiv preprint arXiv:2002.03932* (2020).
- [8] Minmin Chen. 2017. Efficient vector representation for documents through corruption. *arXiv preprint arXiv:1707.02377* (2017).
- [9] Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel S Weld. 2020. Specter: Document-level representation learning using citation-informed transformers. *arXiv preprint arXiv:2004.07180* (2020).
- [10] Nick Craswell, Bhaskar Mitra, Eminie Yilmaz, Daniel Campos, and Ellen M Voorhees. 2020. Overview of the TREC 2019 deep learning track. *arXiv preprint arXiv:2003.07820* (2020).
- [11] Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
- [12] Thomas Diggelmann, Jordan Boyd-Graber, Jannis Bulian, Massimiliano Ciaramita, and Markus Leippold. 2020. Climate-fever: A dataset for verification of real-world climate claims. *arXiv preprint arXiv:2012.00614* (2020).
- [13] Qian Dong, Yiding Liu, Qingyao Ai, Haitao Li, Shuaiqiang Wang, Yiqun Liu, Dawei Yin, and Shaoping Ma. 2023. I3 retriever: incorporating implicit interaction in pre-trained language models for passage retrieval. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 441–451.
- [14] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783* (2024).
- [15] Luyu Gao and Jamie Callan. 2021. Condenser: a pre-training architecture for dense retrieval. *arXiv preprint arXiv:2104.08253* (2021).
- [16] Luyu Gao and Jamie Callan. 2021. Unsupervised corpus aware language model pre-training for dense passage retrieval. *arXiv preprint arXiv:2108.05540* (2021).
- [17] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM international conference on information and knowledge management*. 55–64.
- [18] Faegheh Hasibi, Fedor Nikolaev, Chenyan Xiong, Krisztian Balog, Svein Erik Bratsberg, Alexander Kotov, and Jamie Callan. 2017. DBpedia-entity v2: a test collection for entity search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1265–1268.
- [19] Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. 2021. Efficiently teaching an effective dense retriever with balanced topic aware sampling. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 113–122.
- [20] Doris Hoogeveen, Karin M Verspoor, and Timothy Baldwin. 2015. Cquadupstack: A benchmark data set for community question-answering research. In *Proceedings of the 20th Australasian document computing symposium*. 1–8.
- [21] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685* (2021).
- [22] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2333–2338.
- [23] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118* (2021).
- [24] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lampe, Lucile Saulnier, et al. 2023. Mistral 7B. *arXiv preprint arXiv:2310.06825* (2023).
- [25] Ting Jiang, Shaohan Huang, Zhongzhi Luan, Deqing Wang, and Fuzhen Zhuang. 2023. Scaling sentence embeddings with large language models. *arXiv preprint arXiv:2307.16645* (2023).
- [26] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data* 7, 3 (2019), 535–547.
- [27] Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. 2023. Challenges and Applications of Large Language Models. *CoRR abs/2307.10169* (2023). <https://doi.org/10.48550/ARXIV.2307.10169>
- [28] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906* (2020).
- [29] Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 39–48.
- [30] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics* 7 (2019), 453–466.
- [31] Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024. NV-Embed: Improved Techniques for Training LLMs as Generalist Embedding Models. *arXiv preprint arXiv:2405.17428* (2024).
- [32] Oleg Lesota, Navid Rekabsaz, Daniel Cohen, Klaus Antonius Grasserbauer, Carsten Eickhoff, and Markus Schedl. 2021. A modern perspective on query likelihood with deep generative retrieval models. In *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*. 185–195.
- [33] Chaofan Li, Zheng Liu, Shitao Xiao, Yingxia Shao, and Defu Lian. 2024. Llama2Vec: Unsupervised Adaptation of Large Language Models for Dense Retrieval. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 3490–3500.
- [34] 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*. 2356–2362.
- [35] Sheng-Chieh Lin, Akari Asai, Minghan Li, Barlas Oguz, Jimmy Lin, Yashar Mehdad, Wen-tau Yih, and Xilun Chen. 2023. How to train your dragon: Diverse augmentation towards generalizable dense retrieval. *arXiv preprint arXiv:2302.07452* (2023).
- [36] Yinhai Liu. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).
- [37] Yuxiang Lu, Yiding Liu, Jiaxiang Liu, Yunsheng Shi, Zhengjie Huang, Shikun Feng Yu Sun, Hao Tian, Huo Wu, Shuaiqiang Wang, Dawei Yin, et al. 2022. Ernie-search: Bridging cross-encoder with dual-encoder via self on-the-fly distillation for dense passage retrieval. *arXiv preprint arXiv:2205.09153* (2022).
- [38] Xinyu Ma, Jiafeng Guo, Ruqing Zhang, Yixing Fan, Yingyan Li, and Xueqi Cheng. 2021. B-PROP: bootstrapped pre-training with representative words prediction for ad-hoc retrieval. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1513–1522.
- [39] Xueguang Ma, Liang Wang, Nan Yang, Fury 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*. 2421–2425.
- [40] Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. 2018. Wwww'18 open challenge: financial opinion mining and question answering. In *Companion proceedings of the the web conference 2018*. 1941–1942.
- [41] Gabriel de Souza P Moreira, Radek Osmulski, Mengyao Xu, Ronay Ak, Benedikt Schifferer, and Even Oldridge. 2024. NV-Retriever: Improving text embedding models with effective hard-negative mining. *arXiv preprint arXiv:2407.15831* (2024).
- [42] Niklas Muennighoff. 2022. Sqpt: Gpt sentence embeddings for semantic search. *arXiv preprint arXiv:2202.08904* (2022).
- [43] Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, et al. 2022. Text and code embeddings by contrastive pre-training. *arXiv preprint arXiv:2201.10005* (2022).
- [44] Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernández Ábrego, Ji Ma, Vincent Y Zhao, Yi Luan, Keith B Hall, Ming-Wei Chang, et al. 2021. Large dual encoders are generalizable retrievers. *arXiv preprint arXiv:2112.07899* (2021).
- [45] Rodrigo Nogueira, Jimmy Lin, and AI Epistemic. 2019. From doc2query to docTTTQuery. *Online preprint 6*, 2 (2019).

- [46] Jay M Ponte and W Bruce Croft. 2017. A language modeling approach to information retrieval. In *ACM SIGIR Forum*, Vol. 51. ACM New York, NY, USA, 202–208.
- [47] Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2020. RocketQA: An optimized training approach to dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2010.08191* (2020).
- [48] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deep-speed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3505–3506.
- [49] Ruiyang Ren, Yingqi Qu, Jing Liu, Wayne Xin Zhao, Qiaoqiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. 2021. Rocketqv2: A joint training method for dense passage retrieval and passage re-ranking. *arXiv preprint arXiv:2110.07367* (2021).
- [50] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2021. Colbertv2: Effective and efficient retrieval via lightweight late interaction. *arXiv preprint arXiv:2112.01488* (2021).
- [51] Jacob Mitchell Springer, Suhas Kotha, Daniel Fried, Graham Neubig, and Aditi Raghunathan. 2024. Repetition improves language model embeddings. *arXiv preprint arXiv:2402.15449* (2024).
- [52] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663* (2021).
- [53] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. *arXiv preprint arXiv:1803.05355* (2018).
- [54] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).
- [55] Ellen Voorhees, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, William R Hersh, Kyle Lo, Kirk Roberts, Ian Soboroff, and Lucy Lu Wang. 2021. TREC-COVID: constructing a pandemic information retrieval test collection. In *ACM SIGIR Forum*, Vol. 54. ACM New York, NY, USA, 1–12.
- [56] Henning Wachsmuth, Shahbaz Syed, and Benno Stein. 2018. Retrieval of the best counterargument without prior topic knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 241–251.
- [57] David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. 2020. Fact or fiction: Verifying scientific claims. *arXiv preprint arXiv:2004.14974* (2020).
- [58] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Dixin Jiang, Rangan Majumder, and Furu Wei. 2022. Simlm: Pre-training with representation bottleneck for dense passage retrieval. *arXiv preprint arXiv:2207.02578* (2022).
- [59] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Dixin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533* (2022).
- [60] Shitao Xiao, Zheng Liu, Yingxiao Shao, and Zhao Cao. 2022. RetroMAE: Pre-training retrieval-oriented language models via masked auto-encoder. *arXiv preprint arXiv:2205.12035* (2022).
- [61] Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. 2023. Adaptive Chameleon or Stubborn Sloth: Unraveling the Behavior of Large Language Models in Knowledge Conflicts. *arXiv preprint arXiv:2305.13300* (2023).
- [62] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808* (2020).
- [63] Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the use of lucene for information retrieval research. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*. 1253–1256.
- [64] Zhilin Yang. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv preprint arXiv:1906.08237* (2019).
- [65] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600* (2018).
- [66] Chengxiang Zhai and John Lafferty. 2004. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems (TOIS)* 22, 2 (2004), 179–214.
- [67] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiefeng Guo, Min Zhang, and Shaoping Ma. 2021. Optimizing dense retrieval model training with hard negatives. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1503–1512.
- [68] Hang Zhang, Yeyun Gong, Yelong Shen, Jiancheng Lv, Nan Duan, and Weizhu Chen. 2021. Adversarial retriever-ranker for dense text retrieval. *arXiv preprint arXiv:2110.03611* (2021).
- [69] Shengya Zhuang, Hang Li, and Guido Zuccon. 2021. Deep query likelihood model for information retrieval. In *Advances in Information Retrieval: 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28–April 1, 2021, Proceedings, Part II* 43. Springer, 463–470.