# Dense Text Retrieval Based on Pretrained Language Models: A Survey



WAYNE XIN ZHAO, Renmin University of China, China
JING LIU, Baidu Inc., China
RUIYANG REN and JI-RONG WEN, Renmin University of China, China

Text retrieval is a long-standing research topic on information seeking, where a system is required to return relevant information resources to user's queries in natural language. From heuristic-based retrieval methods to learning-based ranking functions, the underlying retrieval models have been continually evolved with the ever-lasting technical innovation. To design effective retrieval models, a key point lies in how to learn text representations and model the relevance matching. The recent success of pretrained language models (PLM) sheds light on developing more capable text-retrieval approaches by leveraging the excellent modeling capacity of PLMs. With powerful PLMs, we can effectively learn the semantic representations of queries and texts in the latent representation space, and further construct the semantic matching function between the dense vectors for relevance modeling. Such a retrieval approach is called *dense retrieval*, since it employs dense vectors to represent the texts. Considering the rapid progress on dense retrieval, this survey systematically reviews the recent progress on PLM-based dense retrieval. Different from previous surveys on dense retrieval, we take a new perspective to organize the related studies by four major aspects, including architecture, training, indexing and integration, and thoroughly summarize the mainstream techniques for each aspect. We extensively collect the recent advances on this topic, and include 300+ reference papers. To support our survey, we create a website for providing useful resources, and release a code repository for dense retrieval. This survey aims to provide a comprehensive, practical reference focused on the major progress for dense text retrieval.

CCS Concepts: • **Information systems** → **Information retrieval;**

Additional Key Words and Phrases: Text retrieval, dense retrieval, pretrained language models

**ACM Reference format:**
Wayne Xin Zhao, Jing Liu, Ruiyang Ren, and Ji-Rong Wen. 2024. Dense Text Retrieval Based on Pretrained Language Models: A Survey. *ACM Trans. Inf. Syst.* 42, 4, Article 89 (February 2024), 60 pages.
https://doi.org/10.1145/3637870

This work was partially supported by National Natural Science Foundation of China under Grants No. 62222215 and No. U2001212, and Beijing Natural Science Foundation under Grant No. 4222027.
Authors' addresses: W. Xin Zhao and R. Ren, Gaoling School of Artificial Intelligence, Renmin University of China, No. 59 Zhongguancun Street, Haidian District, Beijing, China, 100872; e-mails: batmanfly@gmail.com, reyon. ren@ruc.edu.cn; J. Liu (Corresponding author), Baidu Inc., Beijing, China, 100193; e-mail: liujing46@baidu.com; J.-R. Wen, Gaoling School of Artificial Intelligence, School of Information, Renmin University of China, No. 59 Zhongguancun Street, Haidian District, Beijing, China, 100872; e-mail: jrwen@ruc.edu.cn.

1046-8188/2024/02-ART89 $15.00
https://doi.org/10.1145/3637870

## 1 INTRODUCTION

Text retrieval aims at finding relevant information resources (e.g., documents or passages) in response to user's queries. It refers to a specific information-seeking scenario where the queries and resources are present in the form of natural language text. As one of the most essential techniques to overcome information overload, text-retrieval systems have been widely employed to support many downstream applications, including question answering [111, 270], dialog system [30, 107], entity linking [73, 286], and Web search [183].

The idea of developing text-retrieval systems has a long history in the research literature. Early in the 1950s, pioneering researchers have started to study how to index the texts by selecting representative terms for information retrieval [110]. Among the early efforts, a significant achievement is the *vector space model* [231, 233] based on the "bag-of-words" assumption, representing both documents and queries as sparse term-based vectors. To construct sparse vector representations, various term weighting methods have been designed and implemented, including the classic *tf-idf* method [1, 225, 232]. Based on this scheme, the relevance can be estimated according to the lexical similarity between sparse query and text vectors. Such a representation scheme is further supported by the well-known data structure of *inverted index* [336, 337], which organizes the text content as term-oriented posting lists, for efficient text retrieval. To better understand the underlying retrieval mechanism, probabilistic relevance frameworks have been proposed for relevance modeling, exemplified by the classic BM25 model [226, 227]. Furthermore, statistical language modeling approaches [304] have been widely explored for text ranking. These early contributions lay the foundation of modern information-retrieval systems, while the proposed retrieval methods are usually based on heuristic strategies or simplified probabilistic principles.

With the development of machine learning discipline, *learning to rank* [125, 149] introduces *supervised learning* for text ranking. The basic idea is to design feature-based ranking functions taking as input hand-crafted features (not only limited to lexical features) and then train the ranking function with relevance judgements (binary or graded relevance annotations over documents). Despite the flexibility, learning to rank methods still rely on human efforts for feature engineering.

Further, the re-surge of neural networks sheds lights on developing more capable text-retrieval systems, which no longer require hand-crafted text features. As an important progress in information retrieval, deep learning approaches [96] can learn query and document representations from labeled data in an automatic way, where both queries and documents are mapped into low-dimensional vectors (called *dense vectors* or *embeddings*) in the latent semantic space. In this manner, the relevance can be measured according to the semantic similarity between the dense vectors. In contrast to sparse vectors in the classic vector space model, embeddings do not correspond to explicit term dimensions, but instead aim at capturing latent semantic characteristics for matching. Such a retrieval paradigm is called ***Neural Information Retrieval (Neural IR)*** [75, 77, 182], which can be considered as initial explorations for dense retrieval techniques. Following the convention in References [57, 137], we refer to these neural IR methods at this stage as *pre-BERT models*.

Recently, based on the powerful Transformer architecture [268], **pretrained language models (PLM)** [53] have significantly pushed forward the progress of language intelligence. Pretrained on large-scale general text data, PLMs can encode large amounts of semantic knowledge, thus having an improved capacity to understand and represent the semantics of text content. Following the "*pretrain then fine-tune*" paradigm, PLMs can be further fine-tuned according to specific downstream tasks. Inspired by the success of PLMs in natural language processing, since 2019, researchers have started to develop text-retrieval models based on PLMs, leading to the new generation of text-retrieval approaches, i.e., *PLM-based dense retrieval models*.

In the past four years, a large number of studies on PLM-based dense retrieval have been proposed [22, 57, 137], which have largely raised the performance bar on existing benchmark datasets. It has been reported that PLM-based approaches have dominated the top results for both document and passage retrieval tasks at the 2021 Deep Learning Track [46]. The reason for the success of PLM-based dense retrieval models can be given in two major aspects. First, the excellent text representation capability of PLMs enables the text-retrieval systems to answer difficult queries that cannot be solved via simple lexical match. Second, the availability of large-scale labeled retrieval datasets (e.g., MS MARCO [185] and Natural Questions [116] datasets) makes it feasible to train (or fine-tune) capable PLMs for text retrieval. For example, TREC 2019 and 2020 Deep Learning Track [44, 47] released a training set of 0.367 million queries (derived from MS MARCO [185]) for document retrieval task, which is significantly larger than those in previous retrieval tasks in TREC.

Considering the important progress on dense retrieval, this survey aims to provide a systematic review of existing text-retrieval approaches. In particular, we focus on the PLM-based dense retrieval approaches (short as *dense retrieval models* throughout this survey) instead of previous neural IR models (i.e., the pre-BERT methods [75, 96]). This survey takes *first-stage retrieval* as the core, and extensively discusses four related aspects to build a dense retrieval system, including *architecture* (how to design the network architectures for dense retrievers), *training* (how to optimize the dense retriever with special training strategies), *indexing* (how to design efficient data structure for indexing and retrieving dense vectors), and *integration* (how to integrate and optimize a complete retrieval pipeline). Our survey extensively discusses various useful topics or techniques for building a dense retrieval system, which aims to provide a *comprehensive*, *practical* reference to this research direction for researchers and engineers.

We are aware of several closely related surveys or books on dense retrieval [22, 57, 77, 137, 263]. For example, Guo et al. [77] review early contributions in neural IR, Cai et al. [22] summarize the major progress for the first-stage retrieval in three different paradigms, Lin et al. [137] present a review of text ranking methods mainly based on pretrained Transformers, Fan et al. [57] review the major progress for information retrieval based on pretraining methods, and Nicola Tonellotto [263] present a lecture note of recent progress on neural information retrieval. Different from these literature surveys, we highlight three new features of this survey as follows: (i) We concentrate on the research of *PLM-based dense retrieval*, and organize the related studies by a new categorization in four major aspects, i.e., architecture, training, indexing and integration; (ii) We take a special focus on *practical techniques* for dense retrieval, extensively discussing the approaches to train the retrieval models, to build the dense index, and optimize the retrieval pipeline; (iii) We also discuss several emerging research topics (e.g., model based retrieval), which will complement the content of dense retrieval in alternative approaches or improvement techniques.

To support this survey, we create a referencing website for including related resources (e.g., papers, dataset, and library) on dense retrieval research, at the link: https://github.com/RUCAIBox/DenseRetrieval. Furthermore, we implement and release a code repertory for a number of dense retrieval models, and provide flexible interfaces to use or retrain dense retrievers.

The remainder of the survey is organized as follows: Section 2 provides the overall introduction to the terminology, notations and task settings for dense retrieval. As the core content, Sections 3, 4, 5, and 6 review the progress on the mainstream architecture, training approach, index mechanism and retrieval pipeline for dense retrieval. Then, we continue the discussion on the advanced topics (Section 7) and the applications (Section 8). Finally, we conclude this survey by summarizing the major findings and remaining issues in Section 9.

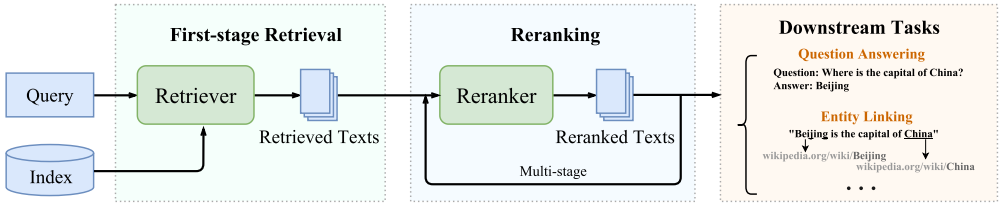



Fig. 1. Illustration for the overall pipeline of an information-retrieval system.

## 2 OVERVIEW

This section first introduces the background about dense text retrieval and then discusses the key aspects for designing the dense retrieval models.

### 2.1 Task Setting, Terminology and Evaluation

In this survey, we focus on the task of finding relevant texts from a large text collection in response to a natural language query issued by a user. Specially, both query and text (e.g., a document) are presented in the form of a sequence of word tokens from a vocabulary. In particular, *texts* can be in different semantic granularities (e.g., document, passage or sentence), leading to different types of retrieval tasks such as document retrieval and passage retrieval. Since we aim to introduce general methodologies for various retrieval tasks or settings, we refer to these tasks as *text retrieval* in a unified way. Typically, a complete information-retrieval system consists of multiple procedures or stages arranged in a processing pipeline [57, 137]. In this pipeline, the first-stage retrieval aims to reduce the candidate space by retrieving relevant candidates, and we refer to the component that implements the first-stage retrieval as *retriever* [22, 137]. Correspondingly, the subsequent stages mainly focus on *reranking* (or *ranking*) the candidate texts, called *reranking stages*, which are supported by the *rerankers* [137]. Based on the retrieval results from the first-stage retriever, a retrieval system usually sets up one or multiple reranking stages to refine the initial results and derive the final search results. For other fine-grained retrieval tasks, e.g., question answering, another component called *reader* [29] may be further integrated to analyze the returned texts by the retriever (or reranker) and locate the answers for the query.

Compared with traditional retrieval models, dense retrieval models are more data hungry, requiring large-scale labeled datasets to learn the parameters of the PLMs. Among the recently released datasets, *MS MARCO* dataset [185] contains a large amount of queries with annotated relevant passages in Web documents, and ***Natural Questions (NQ)*** [116] dataset contains search queries from Google and documents with paragraphs and answer spans from the top-ranked Wikipedia pages. A recent study [48] has summarized the promotion effect of MS MARCO on the progress of dense retrieval: "*The MS MARCO datasets have enabled large-data exploration of neural models.*" Besides, based on MS MARCO, several variants have been created to enrich the evaluation characteristics on some specific aspect, e.g., the multilingual version mMARCO [18] and the MS MARCO Chameleons dataset [3] (consisting of obstinate queries that are difficult to answer by neural retrievers while having similar query length and distribution of relevance judgements with easy queries). Also, Sciavolino et al. [236] create EntityQuestions dataset, as a challenging test set for the models trained on NQ, which contains simple factoid questions about entities from Wikipedia. Further, several comprehensive benchmark datasets are released to evaluate the overall retrieval capability of the retrieval models by aggregating representative datasets and conducting diverse evaluation tasks, such as *BEIR* [260] and *KILT* [198]. Although existing datasets largely improve the training of dense retrievers, in these datasets, a query typically corresponds to very few relevance judgements. Although existing datasets largely improve the training of dense retrievers, in these datasets, a query

typically corresponds to very few relevance judgements. For example, Nogueira et al. observe that most of queries in MS MARCO dataset contains only one labeled positive[4], which is likely to be smaller than the actual number of relevant ones in the collection. Incomplete relevance annotations will lead to several training issues such as false negative, which potentially affects the retrieval performance. Corresponding to these public datasets, a number of open-sourced dense retrieval libraries have been also released, including Tevatron [69], Pyserini [136], SentenceTransformers [215], Asyncval [334], and OpenMatch [155].

To evaluate the retrieval capacity of an information-retrieval system, a number of factors need to be considered [81, 81, 267]: effectiveness, efficiency, diversity, novelty, and so on. Existing studies mainly focus on the effectiveness for the retrieval system, and they widely adopt Top-$K$ Accuracy, Recall, Precision, **Mean Average Precision (MAP)**, **Mean Reciprocal Rank (MRR)**, and **Normalized Discounted Cumulative Gain (NDCG)** as evaluation metrics. Note that these metrics can be used in both first-stage retrieval and reranking. However, we only describe the evaluation metrics from a general ranking perspective without considering specific task scenarios. In particular, top-ranked texts are more important for retrieval tasks, and therefore cut-off metrics are often adopted to examine the quality of texts at top positions.

Due to the space limit, we omit the detailed descriptions of datasets, metrics and tools. And, we provide a comprehensive description of the three parts in the Appendix of this survey.[1]

### 2.2 Formulation for Dense Retrieval

Formally, let $q$ denote a natural language query and $d_i$ denote a text from a large text collection $\mathcal{D} = \{d_i\}_{i=1}^{m}$ consisting of $m$ documents. Given a query, text retrieval aims to return a ranked list of $n$ most relevant texts $\mathcal{L} = [d_1, d_2, \cdots d_n]$ according to the relevance scores by a retrieval model. As the technical approach, either sparse retrieval models (relying on exact lexical matching) or dense retrieval models (relying on semantic matching) can be used to implement the retriever.

The key of dense retrieval lies in the fact that queries and texts are represented by dense vectors, such that the relevance score can be computed according to some similarity function between these dense vectors [22, 57, 137], denoted by

$$\text{Rel}(q, d) = f_{\text{sim}}\Big(\phi(q), \psi(d)\Big), \tag{1}$$

where $\phi(\cdot) \in \mathbb{R}^l$ and $\psi(\cdot) \in \mathbb{R}^l$ are functions mapping queries and texts into $l$-dimensional vectors, respectively. For dense retrieval, $\phi(\cdot)$ and $\psi(\cdot)$ are developed based on neural network encoder, and some measurement function (e.g., inner product) can be used to implement $f_{\text{sim}}(\cdot)$. At the pre-BERT time,[2] the encoders are usually implemented by multi-layered neural networks, while in this survey we focus on the text encoder based on PLMs, called *dense retrievers*.

To learn the dense retrieval models, we also assume that a set of relevance judgements are provided for training or fine-tuning the PLMs. Specially, we only consider the setting with binary (positive only) relevance judgements: for a query $q$, a list of positive texts $\{d_i^+\}$ are given as training data. Usually, negative texts (called *negatives*) are not directly available, and we need to obtain negatives through sampling or other strategies (detailed in Section 4.2). Note that in some cases, we can also obtain graded relevance judgements [104, 246], where a label indicates the relevance degree of a text (e.g., relevant, partially relevant and non-relevant). However, such fine-grained relevance labels are difficult to obtain in real-world retrieval systems. Thus, we mainly consider the binary relevance judgements in this survey.

[1] https://github.com/RUCAIBox/DenseRetrieval/blob/main/Appendix.pdf

[2] Following Reference [189], we refer to the neural information-retrieval methods as *pre-BERT approaches* before the use of PLMs.

### 2.3 Key Aspects

This survey takes the *first-stage dense retrieval* as the core, and extensively discusses four key aspects for building a capable retrieval system, which are detailed as follows:

— *Architecture* (Section 3). It is important to design suitable network architectures for building the relevance model. Currently, a number of general-purpose PLMs have been developed, and we need to adapt existing PLMs to text retrieval according to the task requirement.
*Training* (Section 4). Different from traditional retrieval models, it is more difficult to train PLM-based retrievers, consisting of a huge number of parameters. We need to develop effective training approaches to achieving desirable retrieval performance.
— *Indexing* (Section 5). Traditionally, inverted index can efficiently support the sparse retrieval models. However, dense vectors do not explicitly correspond to lexical terms. We need to design suitable indexing mechanism for dense retrieval.
— *Integration* (Section 6). As mentioned before, retrieval systems are usually implemented via a pipeline consisting of retrievers and rerankers. It is necessary to study how to integrate and optimize the dense retriever in a whole retrieval pipeline.

## 3 ARCHITECTURE

This section describes the major network architectures for dense text retrieval. We start with the background of Transformer and PLMs, then introduce two typical neural architectures for building dense retrieval models, and finally compare sparse and dense retrieval models.

### 3.1 Background

*3.1.1 Transformer and Pretrained Language Models.* Transformer [268] has become the mainstream backbone for language model pretraining, which was originally proposed for efficiently modeling sequence data. Different from traditional sequence neural networks (i.e., RNN and its variants [40, 88]), Transformer no longer processes the data in order, but instead introduces a new *self-attention* mechanism: a token attends to all the positions from the input. Such an architecture is particularly suited to be parallelized with the hardware support, making it feasible to train very large neural networks. Based on Transformer and its variants, PLM are proposed [21, 53, 152] in the field of natural language processing. PLMs are pretrained over a large-scale general text corpus with specially designed self-supervised loss functions, and can be further fine-tuned according to downstream tasks, called *pretraining then fine-tuning paradigm* [53, 148], which is one of the most striking achievements on language intelligence in recent years. As one of the most representative PLMs, BERT [53] pretrains deep bidirectional architectures for learning general text representations (with the trick of word masking), which has largely raised the performance bar in a variety of natural language processing tasks. With excellent representation capacities, PLMs also shed lights on developing more effective retrieval models.

*3.1.2 PLMs for Sparse Retrieval.* Before introducing the use of PLMs for dense retrieval, this part briefly reviews how to utilize PLMs for improving sparse retrieval models that rely on lexical match. Basically speaking, this line of research work can be roughly divided into two major categories, namely, term weighting and term expansion.

The first category of work aims to improve term weighting based on per-token contextualized representations. As a representative study, DeepCT [51] utilizes the learned BERT token representations for estimating context-specific importance of an occurring term in each passage. The basic idea is to regress the token representations into real-valued term weights. HDCT [50] extends it to long documents by splitting the documents into passages, and aggregates the estimated term weights in each passage. Specifically, it leverages three kinds of data resources (i.e., titles,

Table 1. Detailed List of Different Dense Retrieval Methods in the Literature with Detailed Configurations

| Years | Methods | PLMs | Arch. | Training | | | | Other Tricks |
|---|---|---|---|---|---|---|---|---|
| | | | | Loss | Negative | Data Aug. | Pretrain | |
| 2020 | Poly-encoders [98] | $\text{BERT}_{\text{base}}$ | MR | CE | | | TAP | Context-candidate attention |
| 2019 | ICT [120] | $\text{BERT}_{\text{base}}$ | SR | IOC | | | TAP | Joint training |
| 2020 | ICT+BFS+WLP [28] | $\text{BERT}_{\text{base}}$ | SR | IOC | | | TAP | |
| 2020 | REALM [79] | $\text{BERT}_{\text{base}}$ | SR | IOC | | | RAP | Joint training + Async. index |
| 2020 | DPR [111] | $\text{BERT}_{\text{base}}$ | SR | NLL | IB+SHN | ALD | | |
| 2020 | ColBERT [113] | $\text{BERT}_{\text{base}}$ | MR | CE | | | | Multi-core pre-processing |
| 2020 | ME-BERT [161] | $\text{BERT}_{\text{large}}$ | MR | CE | IB+DyHN | | | Sparse-dense hybrid |
| 2020 | RepBERT [309] | $\text{BERT}_{\text{base}}$ | SR | IOC | IB | | | |
| 2020 | ANCE [289] | $\text{RoBERTa}_{\text{base}}$ | SR | NLL | DyHN | | | Async. index |
| 2020 | EZRQG [135] | $\text{BERT}_{\text{base}}$ | SR | IOC | | | | Query generation |
| 2020 | AugmentedBERT [297] | $\text{BERT}_{\text{base}}$ | SR | IOC+KL | SHN | KD | | Kernel density estimation |
| 2020 | In-batch KD [89] | $\text{BERT}_{\text{base}}$ | SR | MSE | IB | KD | | Cross-architecture KD |
| 2020 | RocketQA [207] | $\text{ERNIE}_{\text{base}}$ | SR | NLL | CB+DeHN | KD | | Iter. training |
| 2020 | AugmentSBERT [259] | $\text{BERT}_{\text{base}}$ | SR | NLL | SHN+DeHN | KD | | Kernel density estimation |
| 2020 | TCT-ColBERT [141] | $\text{BERT}_{\text{base}}$ | SR | KL | IB+SHN | KD | | Sparse-dense hybrid |
| 2020 | Multi-stage [157] | $\text{BERT}_{\text{base}}$ | SR | NLL | IB+SHN | KD | TAP | Embedding fusion |
| 2020 | DKRR[102] | $\text{BERT}_{\text{base}}$ | SR | KL | SHN | KD | | Iter. training |
| 2020 | DensePhrases [118] | $\text{SpanBERT}_{\text{base}}$ | PR | IOC+KL | IB | KD | | Pre-batch negatives |
| 2020 | UnifiedQA [192] | $\text{BERT}_{\text{base}}$ | SR | NLL | IB | ALD | | |
| 2021 | Individual Top-k [229] | $\text{BERT}_{\text{large}}$ | SR | IOC+NLL | IB+SHN | | TAP | Joint training |
| 2021 | Grad. Cache [70] | $\text{BERT}_{\text{base}}$ | SR | NLL | CB | | | Gradient caching |
| 2021 | TAS-Balanced [92] | $\text{BERT}_{\text{base}}$ | SR | MSE | IB+SHN | KD | | Balanced topic aware sampling |
| 2021 | ADORE+STAR [308] | $\text{RoBERTa}_{\text{base}}$ | SR | NLL | SHN+DyHN | | | Async. index |
| 2021 | Condenser [65] | Condenser | SR | CE | IB | | REP | |
| 2021 | DRPQ [256] | $\text{BERT}_{\text{base}}$ | MR | NLL | IB+DyHN | | | Multiple pseudo query embeddings |
| 2021 | RANCE [202] | $\text{RoBERTa}_{\text{base}}$ | SR | NLL | DeHN | | | |
| 2021 | DANCE [132] | $\text{RoBERTa}_{\text{base}}$ | SR | IOC | SHN | | | Contrastive dual learning |
| 2021 | DPR-PAQ [193] | $\text{RoBERTa}_{\text{large}}$ | SR | NLL | IB+SHN | ALD | GAP | |
| 2021 | JPQ [306] | $\text{BERT}_{\text{base}}$ | SR | IOC | DyHN | | | Quantization |
| 2021 | coCodenser [66] | Condenser | SR | NLL | SHN | | REP | Corpus-aware pretraining |
| 2021 | PAIR [218] | $\text{ERNIE}_{\text{base}}$ | SR | IOC+NLL | IB+SHN | KD | REP | Passage-centric constrain |
| 2021 | ST5 [186] | $\text{T5}_{\text{11B}}$ | SR | NLL | IB | | | |
| 2021 | ANCE-PRF [301] | $\text{RoBERTa}_{\text{base}}$ | SR | NLL | DyHN | | | Pseudo relevance feedback |
| 2021 | Trans-Encoder [144] | $\text{RoBERTa}_{\text{base}}$ | SR | MSE | DyHN | KD | | Iter. training + Mutual distillation |
| 2021 | AR2-G [311] | $\text{ERNIE}_{\text{base}}$ | SR | IOC | DyHN | | | Adversarial training |
| 2021 | RepCONC [307] | $\text{BERT}_{\text{base}}$ | SR | IOC | DyHN | | | Quantization |
| 2021 | RocketQAv2 [219] | $\text{ERNIE}_{\text{base}}$ | SR | KL | SHN+DeHN | KD | | Joint training + Mutual distillation |
| 2021 | SPAR [37] | $\text{BERT}_{\text{base}}$ | SR | NLL | SHN | KD | | |
| 2021 | SEED [159] | $\text{BERT}_{\text{base}}$ | SR | NLL | DyHN | | TAP | Autoencoder pretraining |
| 2021 | ColBERTv2 [235] | $\text{BERT}_{\text{base}}$ | SR | NLL | DeHN | KD | | Residual compression + Quantization |
| 2021 | GPL [277] | $\text{DistilBERT}_{\text{base}}$ | SR | MSE | IB+SHN | KD | TAP | Query generation |
| 2021 | Spider [213] | $\text{BERT}_{\text{base}}$ | SR | NLL | IB+SHN | | TAP | Sparse-dense hybrid |
| 2021 | DrBoost [122] | $\text{BERT}_{\text{base}}$ | SR | NLL | SHN | | | Boosting |
| 2021 | GTR [187] | $\text{T5}_{\text{11B}}$ | SR | NLL | IB+DeHN | | TAP | |
| 2021 | Contriever [101] | $\text{BERT}_{\text{base}}$ | SR | NLL | IB | | REP | Contrastive learning |
| 2022 | Uni-Retriever [313] | $\text{BERT}_{\text{base}}$ | SR | IOC+NLL | CB + SHN | | | |
| 2022 | LaPraDoR [290] | $\text{DistilBERT}_{\text{base}}$ | SR | IOC+NLL | IB | | REP | Sparse-dense hybrid |
| 2022 | HLP [322] | $\text{BERT}_{\text{base}}$ | SR | NLL | IB | | TAP | |
| 2022 | DAR [106] | $\text{RoBERTa}_{\text{large}}$ | SR | NLL | IB+SHN | | | Interpolation and perturbation |
| 2022 | MVR [316] | $\text{BERT}_{\text{base}}$ | MR | IOC+NLL | SHN | | | Multi-view representation |
| 2022 | ColBERTer [91] | $\text{DistilBERT}_{\text{base}}$ | MR | IOC+MSE | SHN | | | |
| 2022 | CharacterBERT [335] | CharacterBERT | SR | NLL+KL | IB+SHN | | | Self-teaching |
| 2022 | GNN-encoder [146] | $\text{ERNIE}_{\text{base}}$ | SR | IOC+NLL | IB+DeHN | KD | | Query-passage interaction |
| 2022 | COSTA [164] | $\text{BERT}_{\text{base}}$ | SR | NLL | SHN | | TAP | Contrastive span prediction |
| 2022 | CL-DRD [303] | $\text{DistilBERT}_{\text{base}}$ | SR | KL | KD | | | Curriculum Learning |
| 2022 | ERNIE-Search [160] | $\text{ERNIE}_{\text{large}}$ | SR | NLL+KL | IB | KD | | Joint training + Mutual distillation |
| 2022 | RetroMAE [154] | $\text{BERT}_{\text{base}}$ | SR | IOC+NLL | DyHN | KD | REP | MAE pretraining |
| 2022 | DCSR [93] | $\text{BERT}_{\text{base}}$ | MR | NLL | IB+SHN | | | |
| 2022 | ART [230] | $\text{BERT}_{\text{large}}$ | SR | KL | | | | Query reconstruction |
| 2022 | SimLM [279] | $\text{BERT}_{\text{base}}$ | SR | NLL | IB+DeHN | KD | REP | Iter. training |
| 2022 | RoDR [38] | $\text{BERT}_{\text{base}}$ | SR | NLL+KL | IB+SHN | | | Passage-centric constraint |
| 2022 | Aggretriever [139] | Condenser | SR | NLL | IB+SHN | | | Text aggregation |
| 2022 | CoT-MAE [287] | $\text{BERT}_{\text{base}}$ | SR | NLL | SHN | | GAP | |
| 2022 | CPDAE [167] | $\text{BERT}_{\text{base}}$ | SR | IOC+NLL | SHN | | TAP | Autoencoder pretraining |
| 2022 | DPTDR [257] | $\text{RoBERTa}_{\text{large}}$ | SR | NLL | IB+DeHN | KD | TAP | Deep prompt tuning |
| 2022 | LED [314] | Condenser | SR | IOC+NLL | IB+SHN | KD | | Rank-consistent regularization |
| 2022 | LexMAE [240] | $\text{BERT}_{\text{base}}$ | SR | IOC+NLL+KL | SHN | KD | REP | MAE pretraining |
| 2022 | Promptagator [52] | FLAN | SR | NLL | IB+SHN | | | Prompt-based query generation |
| 2022 | PROD [142] | $\text{ERNIE}_{\text{base}}$ | SR | NLL+KL | IB+SHN | KD | | Curriculum learning |
| 2022 | TASER [39] | Condenser | SR | NLL | IB+SHN | | | Mixture-of-experts |

Abbreviations are defined in Table 2.

Table 2. Abbreviations for Different Techniques or Strategies

| | Type | | | Abbr |
|---|---|---|---|---|
| Architecture | Single-representation (Section 3.2.2)<br>Multi-representation (Section 3.2.2)<br>Phrase-level representation (Section 3.2.2) | | | SR<br>MR<br>PR |
| Training | Loss | Negative log-likelihood loss (Section 4.1.1)<br>Cross-entropy loss (Section 4.1.1)<br>Triplet ranking loss (Section 4.1.1)<br>Incorporating optimization constraints (Section 4.1.2)<br>MSE loss (Section 4.4)<br>KL-divergence loss (Section 4.4) | | NLL<br>CE<br>TR<br>IOC<br>MSE<br>KL |
| | Negative selection | In-batch negatives (Section 4.2.1)<br>Cross-batch negatives (Section 4.2.2) | | IB<br>CB |
| | | Hard negatives | Static hard negatives (Section 4.2.3)<br>Dynamic hard negatives (Section 4.2.3)<br>Denoised hard negatives (Section 4.2.3) | SHN<br>DyHN<br>DeHN |
| | Data augmentation | Auxiliary and synthetic data enrichment (Section 4.3)<br>Knowledge distillation (Section 4.4) | | ALD<br>KD |
| | Pretraining | Generation-augmented pretraining (Section 4.3.2)<br>Task adaptive pretraining (Section 4.5.1)<br>Retrieval-augmented pretraining (Section 4.5.2)<br>Representation enhanced pretraining (Section 4.5.3) | | GAP<br>TAP<br>RAP<br>REP |

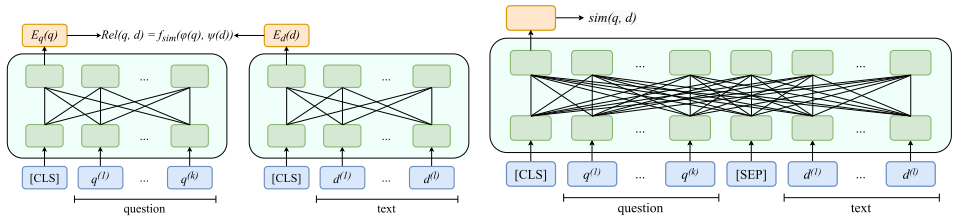
Web inlinks, and pseudo-relevance feedback) to generate weak supervision signals for learning term weights. It can better learn the importance of a term by modeling its text context. Further, COIL [67] utilizes the contextualized token representations of *exact matching terms* for estimating the relevance. It computes the dot product between the token representations from the query encoder and text encoder (only considering the overlapping terms), and then sums the term-specific similarity scores as the relevance score (also including the semantic matching based on "[CLS]" embeddings). To better understand the above models, Lin et al. [138] propose a conceptual framework to unify these approaches: DeepCT aims to assign *scalar weights* to query terms, while COIL aims to assign *vector weights* to query terms. Furthermore, uniCOIL [138] is also proposed by reducing the weight vector in COIL to one dimension, and the experiments in Reference [138] show that uniCOIL can retain the effectiveness while increasing the efficiency.

The second category of work expands queries or documents by using PLMs to mitigate the vocabulary mismatching problem. For example, docTTTTTquery [190] predicts a set of queries that a document will be relevant to, such that the predicted queries can be used to enrich the document content. In addition, SPLADE [63] and SPLADEv2 [62] project each term in queries and texts to a vocabulary-sized weight vector, in which each dimension represents the weight of a term in the vocabulary of PLMs. The weights are estimated by using the logits of masked language models. Then, the final representation of a whole text (or query) is obtained by combining (e.g., sum or max pooling) the weight vectors estimated by all the tokens of the text. Such a representation can be viewed as an expansion of a query (or a text), since it contains the terms that do not occur in the query (or the text). A sparsity regularization is further applied to obtain the sparse representation, to efficiently use the inverted index.

Despite the use of PLMs, these approaches are still based on lexical matching. They can reuse the traditional index structure by incorporating additional payloads (e.g., contextualized embeddings [67]). For a more thorough discussion of PLM-enhanced sparse retrieval models, the readers can refer to References [22, 57, 137].

### 3.2 Neural Architecture for Dense Retrieval

The essence of dense retrieval is to model the *semantic interaction* between queries and texts based on the representations learned in latent semantic space. Based on different interaction modeling




(a) Dual-encoder architecture (b) Cross-encoder architecture

Fig. 2. Illustration of dual-encoder and cross-encoder architectures.

ways, there are two mainstream architectures for dense retrieval, namely, the cross-encoder and bi-encoder. We next introduce the two kinds of major architectures.

*3.2.1 Cross-encoder Architecture.* As a direct application of PLMs, cross-encoder considers a query-text pair as an entire "sentence." To be specific, the input of cross-encoder is the concatenation of a query and a text, separated by a special symbol "[SEP]." To obtain the overall representation, another symbol "[CLS]" is inserted at the beginning of the concatenated sentence. Then, the query-text sequence is fed into a PLM for modeling the semantic interaction between any two tokens of the input sequence. After the representations for each token in the sequence have been learned, we can obtain the match representation for this query-text pair. A commonly used way is to take the "[CLS]" representation as the semantic matching representation [203]. Other variants can be also used, e.g., averaging token embeddings [215]. Such an architecture is similar to *interaction-based architecture* in the pre-BERT studies [75, 288], since it allows the tokens to interact across queries and texts.

*3.2.2 Bi-encoder Architecture.* The bi-encoder (a.k.a., dual-encoder) adopts the two-tower architecture, which is similar to representation-based approaches (e.g., DSSM [96]) in pre-BERT studies. The major difference is that it replaces the previously used multi-layered perceptions (or other neural networks) with PLM-based encoders. Next, we will discuss the major progress of bi-encoder architecture for dense retrieval.

**Single-representation bi-encoder.** As the basic form of bi-encoder architecture, it first learns latent semantic representations for both query and text with two separate encoders, called *query embedding* and *text embedding*, respectively. Then, the relevance score can be computed via some similarity function (e.g., cosine similarity and inner product) between the query embedding and text embedding. As mentioned before, we can directly encode the query and text by placing a special symbol (e.g., "[CLS]" in BERT) at the beginning of a text sequence, so that the learned representation of the special symbol can be used to represent the semantics of a text (or query). Most of the single-representation dense retrievers [111, 207, 289] learn the query and text representations with encoder-only PLMs (e.g., BERT [53], RoBERTa [53], and ERNIE [253]). More recently, **text-to-text Transformer (T5)** [210], which is an encoder-decoder based PLM, has been explored to learn text representations for dense retrieval [186, 187]. It has been shown that a T5-based sentence embedding model outperforms Sentence-BERT [216] on SentEval [42] and SentGLUE [210] tasks.

**Multi-representation bi-encoder.** A major limitation of single-representation bi-encoder is that it cannot well model fine-grained semantic interaction between query and text. To address this issue, several researchers propose to explore multiple-representation bi-encoder for enhancing the text representation and semantic interaction. The poly-encoder model [98] learns $m$ different

representations for modeling the semantics of a text in multiple views, called *context codes* in the original paper. During query time, by attending to the query vector, these $m$ representations are then combined into a single vector. Finally, the inner product between the query vector and aggregated vector is computed as the relevance score. As another related study, ME-BERT [161] also generates $m$ representations for a candidate text, by directly taking the contextualized representations of the first $m$ tokens. The text relevance is computed using the maximum inner product between the query vector and the $m$ contextual representations. Furthermore, ColBERT [113] designs an extreme multi-representation semantic matching model, where per-token contextualized representations are kept for query-text interaction. Different from the representation scheme of cross-encoder (Section 3.2.1), these contextualized representations do not directly interact across queries and texts during encoding. Instead, they introduce a *query-time* mechanism for per-token representation interaction, called *late interaction* [113]. As the extension of ColBERT, ColBERTer [91] designs an approach by combining single-representation (the "[CLS]" embedding) and multi-representation (per-token embeddings) mechanisms for matching, achieving performance improvement over ColBERT.

These multi-representation bi-encoders share the idea of using multiple contextual embeddings for representing queries or texts, such that the similarity can be measured from different semantic views. These contextualized representations are learned and stored during training and indexing; while at query time, we can model fine-grained semantic interaction between query embeddings and pre-computed text embeddings. Such an approach is effective to improve the retrieval performance, but causes a significantly high cost to maintain the multi-representation implementation (e.g., increased index size), which is not practical in real-world applications. Considering this issue, some specific strategies proposed in ColBERTer [91] can be used to reduce the multi-representation costs, such as embedding dimension reduction, bag of unique whole-word representations and contextualized stopword removal. The experiments in Reference [91] demonstrate that ColBERTer can significantly reduce the space cost while retaining the effectiveness.

**Phrase-level representation.** Generally, dense retrieval models focus on solving document- or paragraph-level text-retrieval tasks. It is natural to extend existing retrieval models by considering more fine-grained retrieval units (e.g., phrases or contiguous text segments) for specific tasks, such as open-domain question answering and slot filling [118]. Recently, several studies propose to learn phrase-level representations for directly retrieving phrases as the answers to queries [118, 119, 237, 238]. The basic idea is to preprocess all the documents in a query-agnostic way and generate phrase-level representations (called *phrase embeddings*) for contiguous text segments in the documents. Then, the answer finding task is cast into the problem of retrieving the nearest phrase embeddings to the query embedding. PIQA (phrase-indexed question answering) [237] presents the first approach that leverages dense phrase index for question answering. DenSPI [238] further combines dense and sparse vectors of phrases to capture both semantic and lexical information, to improve the performance of phrase-indexed question answering. While, such an approach heavily relies on sparse vectors for achieving good performance. Considering this issue, Seo et al. [118] propose DensePhrases, an approach for improving the learning of dense phrase encoder without using sparse representations. DensePhrases learns dense phrase representations by data augmentation and knowledge distillation, and further employs enhanced negative training (considering both in-batch and pre-batch negatives) and query-side fune-tuning. An interesting observation found in Reference [119] is that a dense phrase retrieval system can achieve a better passage retrieval accuracy than DPR on NQ and TriviaQA, without any re-training. It shows that phrase-level information is useful for relevance matching by capturing fine-grained semantic characteristics. They further propose that dense phrase retriever can be considered as a

dynamic version of multi-representation retriever, since it dynamically generates a set of phrase embeddings for each text.

*Other improvement variants.* Besides, several studies also attempt to improve these basic approaches, including composite architectures that combine different dense retrieval components [122, 129, 150], lightweight architectures that prune or compress the retrieval models [235, 264, 306], feedback enhanced models that leverage (pseudo)relevance feedback signals [126, 127, 280, 281, 301, 331], and parameter-efficient tuning that efficiently adapts PLMs to the retrieval task [254, 257].

*3.2.3 Comparison between Cross- and Bi-encoder.* Ever since the pre-BERT age, *interaction-based* or *representation-based* approaches have been proposed as the two major architectures for neural IR [75], according to whether fined-grained tokens can interact across queries and texts. Following this convention, cross-encoder and bi-encoder can be understood as PLM-based instantiations for the two kinds of architectures, respectively. To have a deeper understanding, we next compare cross-encoder and bi-encoder from the perspective of dense text retrieval.

First, cross-encoder is more capable of learning fine-grained semantic interaction information for the query-text pair. It is widely recognized that cross-encoder is more effective in relevance matching [188, 207, 297]. In contrast, bi-encoder (in an original implementation) cannot capture the fine-grained interaction between query and text representations. As discussed above, a number of studies aim to improve the representation capacity of bi-encoder by using multiple context embeddings [98, 113]. According to the reported results on MS MARCO [219, 235], the multi-representation bi-encoder performs better than the single-representation bi-encoder, but still worse than the cross-encoder, since it attempts to mimic the cross-encoder in modeling fine-grained semantic interaction. Second, bi-encoder is more flexible and efficient in architecture. For flexibility, it can employ different encoding networks for encoding queries and texts, which allows more flexible architecture design for dense retrieval, e.g., phrase index [238]. For efficiency, it can be accelerated via approximate nearest-neighbor search for large-scale vector recall [108], which is particularly important for practical use of dense retrieval approaches.

Considering the pros and cons, they are often jointly used in retrieval systems for a trade-off between effectiveness and efficiency. Typically, bi-encoder is used for large-scale candidate recall (i.e., first-stage retrieval), and cross-encoder is adopted to implement the reranker or reader (Section 6.1.2). Besides, cross-encoder is often utilized to improve the bi-encoder, e.g., knowledge distillation [219, 259] (Section 4.4) and pseudo labeled data generation [207, 218] (Section 4.3.2).

### 3.3 Connections between Dense Retrieval and Sparse Retrieval

In this part, we first discuss the difference/connection between sparse and dense retrieval, and then introduce how to combine both kinds of approaches for retrieval.

*3.3.1 Discussions.* To understand how dense retrieval models behave and their connections with sparse retrieval models, we present some discussions by focusing on the following questions.

— [*A*] What are the respective strengths and weaknesses of dense and sparse retrieval models?

In the early literature of dense retrieval [111, 207, 289], a number of experimental studies have reported that dense retrieval models often outperform classic sparse models (e.g., BM25), especially on the benchmarks (e.g., MS MARCO) that were originally designed for question answering. Since these benchmarks focus on complex query solving, deep semantic matching capabilities are required to resolve queries containing few overlapping terms with the answers. Take an example query that cannot be resolved by a classic sparse retriever from the DPR paper [111]: Given the question "*Who is the bad guy in lord of the rings?*" a dense retriever can retrieve a relevant text span

"*Sala Baker is best known for portraying the villain Sauron in the Lord of the Rings trilogy*" by capturing the semantic relatedness between "*bad guy*" and "*villain*" [111]. To resolve the term mismatching issue in traditional retrievers, several improved approaches (exemplified by SPLADE [62, 63]) learn more effective sparse representations by allowing term expansion, and these improved sparse retrieval models can also achieve impressive performance on retrieval benchmarks.[3] In contrast with dense retrieval models, sparse retrieval models usually consume less memory [240], and can also leverage classic inverted index structure for efficient retrieval.

A major limitation of dense retrievers is that they rely on labeled data for learning the model parameters. It has been reported that dense retrieval models might perform worse than sparse retrieval models under the *zero-shot* setting (without relevance judgement data from the target domain) on the test datasets from the BEIR benchmark [260]. The conducted experiments on BEIR show that the zero-shot retrieval capacity of dense retrieval models is highly limited compared to classic retrieval models such as BM25 [260]. Hence, a number of studies are developed to improve the zero-shot retrieval capabilities of dense retrievers (see Section 7.1 for a detailed discussion). Besides, sparse models are more capable in solving the queries that require exact match (e.g., keyword or entity retrieval), while dense retrievers seem to perform worse in these cases, especially when the entities or compositions are rare or do not appear in the training set [147, 236]. For example, the DPR paper [111] shows that it fails to capture the entity phrase "*Thoros of Myr*" for the query "*Who plays Thoros of Myr in Game of Thrones?*" while BM25 succeeds in this case. Such a finding is indeed consistent with the underlying mechanism of the two kinds of technical approaches: sparse retrievers are based on lexical matching, while dense retrievers employ latent embeddings for semantic matching, which may result in the loss of salient information due to the compressed semantic representations. Section 3.3.2 will discuss how to enhance the exact match capacity of dense retrieval models by combining sparse and dense retrievers.

— [*B*] Is there any theoretical analysis on the capacity of dense representations on relevance matching, especially how they mimic and relate to sparse retrieval? Is it possible to apply the traditional IR axioms to dense retrieval?

Sparse retrieval builds the lexical matching model based on term based representations, which has a good interpretability in retrieval behaviors. In contrast, dense retrieval employs latent semantic representations for relevance matching, and thus performs better on complex queries but worse on exact match queries. It is important to understand the retrieval behavior of dense retrieval models and their connection with sparse retrieval models (e.g., how dense retrieval models perform exact match). For this purpose, Luan et al. [161] investigate the effect of the embedding size on the ability to mimic sparse retrieval (e.g., bag-of-words models). They demonstrate that the embedding size of dense retrievers should be increased to achieve comparable retrieval performance of bag-of-words models, when the document length increases. In addition, another related study shows that corpus scale has a larger effect on dense retrieval [217]: "the performance for dense representations can decrease quicker for increasing index sizes than for sparse representations." They explain this finding based on a proof that the probability for false positives becomes larger when the index size increases, especially with a decreasing dimensionality.

Another possible approach to understanding the behavior of dense retrievers is *axiomatic analysis* from classic IR literature [58–60]. Specifically, IR axioms are formalized as a set of relevance constraints that reasonable IR models are desired to satisfy or at least partially satisfy [59, 228], e.g., a document containing more occurrences of a query term should receive a higher score, which can provide both interpretable evidence and learning guidance for traditional IR models. However,

[3] https://europe.naverlabs.com/research/machine-learning-for-robotics/splade-models/

it has been found that existing IR axioms may only partially explain [269] or even not be suitable to analyze [24] the behavior of PLM-based IR models. Some further analysis [169] also shows that neural ranking models have different characteristics compared with sparse ranking models. For example, dense models are easier to be affected by adding non-relevant contents, and different ranking architectures based on the same language model may lead to varied retrieval behaviors. Although BERT-based ranking models are not well aligned with traditional IR axioms, recent research [64] also shows that some dense retrieval model (e.g., ColBERT [113]) is aware of term importance in text representations, and can also implicitly mimic exact match for important terms.

Overall, theoretical exploration of dense retrieval still remains to be an open research direction. More efforts are required to understand the relevance matching behaviors of dense retrieval models.

*3.3.2 Combining Sparse and Dense Retrieval models.* In practice, the two kinds of retrieval models can benefit each other by leveraging complementary relevance information. For example, sparse retrieval models can be utilized to provide initial search results for training dense retrieval models [111, 156], while PLMs can be also used to learn term weights to improve sparse retrieval [50, 51] (as discussed in Section 3.1.2). In particular, it also shows that sparse retrieval models can be utilized to enhance the zero-shot retrieval capacity of dense retrieval models (detailed in Section 7.1).

Considering different relevance characteristics captured by both approaches, there are increasing studies that develop a hybrid of sparse and dense retrieval models. A straightforward approach is to aggregate the retrieval scores of the two approaches [111, 161]. Further, a classifier is employed to select among sparse, dense, or hybrid retrieval strategies specially for each individual query[5], which aims to balance the cost and utility of retrievers.

However, these hybrid approaches have to maintain two different indexing systems (inverted index and dense vector index), which is too complicated to be deployed in real practice. To address this issue, Lin et al. [139, 140] propose to learn low-dimensional dense lexical representations by densifying high-dimensional sparse lexical representations, making it feasible to end-to-end learn lexical-semantic combined retrieval systems. The basic idea of densification is to first divide the high-dimensional lexical representations into slices and reduce each sliced representation by specifically designed pooling approaches. Then, these representations are concatenated as the dense lexical representation. For text ranking, the dense lexical representations and the dense semantic representations (i.e., the representation of "[CLS]") are combined for computing similarity scores. The experiment results in References [139, 140] demonstrate that this approach can largely improve the retrieval effectiveness by combining the benefits of both kinds of representations, showing a superior performance on zero-shot evaluation. Furthermore, Chen et al. [37] propose to learn a dense lexical retriever from the weakly supervised data constructed by BM25, so that the learned dense lexical retriever can imitate the retrieval behavior of the sparse retrievers. In this approach, the representations from the dense lexical retriever and dense semantic retriever (e.g., DPR) are combined for relevance computation. It has been shown that the knowledge learned from BM25 can help the dense retriever on queries containing rare entities, to improve the robustness [37].

## 4 TRAINING

After introducing the network architecture, we next discuss how to effectively train PLM-based dense retrievers, which is the key to achieve good retrieval performance. Focused on the training of bi-encoder for first-stage retrieval, we will first formulate the loss functions and introduce three major issues for training (i.e., large-scale candidate space, limited relevance judgements and pretraining discrepancy), and then discuss how to address the three issues, respectively.

### 4.1 Formulation and Training Issues

This section presents the formulation and issues for training the PLM-based dense retrievers.

*4.1.1 Loss Function.* In this part, we first formulate the retrieval task as a learning task via the *negative log-likelihood loss*, and then discuss several variants to implement the loss functions.

**Negative log-likelihood loss.** Following the notations introduced in Section 2.2, we assume that a set of binary positive relevance judgements (e.g., like or click) are given as supervision signals, denoted by $\mathcal{R} = \{\langle q_i, d_i^+ \rangle\}$, where $d_i^+$ denotes a relevant document for query $q_i$. To optimize the dense retrievers, the core idea is to maximize the probabilities of the relevant texts w.r.t. queries. For simplicity, we introduce the negative log-likelihood loss for a query-positive pair as follows:

$$\mathcal{L}\left(\langle q_i, d_i^+ \rangle\right) = -\log \frac{e^{f(\phi(q_i), \psi(d_i^+))}}{e^{f(\phi(q_i), \psi(d_i^+))} + \sum_{d' \in \mathcal{D}^-} e^{f(\phi(q_i), \psi(d'))}}, \tag{2}$$

where $f(\cdot)$ is a similarity function (often set to the dot product) that measures the similarity between query embedding $\phi(q_i)$ and text embedding $\psi(d_i^+)$, and $\phi(\cdot)$ and $\psi(\cdot)$ are the query encoder and text encoder (Equation (1)), respectively. Here, we enumerate all the text candidates in the collection $\mathcal{D}$ except the positives for computing the exact likelihood. Futher, a negative sampling trick is usually adopted to reduce the computational cost [111, 207]:

$$\mathcal{L}\left(\langle q_i, d_i^+ \rangle\right) = -\log \frac{e^{f(\phi(q_i), \psi(d_i^+))}}{e^{f(\phi(q_i), \psi(d_i^+))} + \sum_{d' \in \mathcal{N}_{q_i}} e^{f(\phi(q_i), \psi(d'))}}, \tag{3}$$

where we sample or select a small set of negative samples for query $q_i$, denoted by $\mathcal{N}_{q_i}$. The above training objective advocates to increase the likelihood of positive texts and decrease the likelihood of sampled negative ones. Actually, such an optimization objective is similar to the InfoNCE loss [266] from contrastive learning, where the loss is constructed by contrasting a positive pair of examples against a set of random example pairs.

**Other loss functions.** Instead of optimizing over a set of negatives, triplet ranking loss [55, 111] optimizes the difference between a positive and a negative given a query, which is defined as

$$\mathcal{L}\left(\langle q_i, d_i^+, d_i^- \rangle\right) = \max\left(0, 1 - (f(\phi(q_i), \psi(d_i^+)) - f(\phi(q_i), \psi(d_i^-)))\right).$$

It has been reported that negative log-likelihood loss is better than the triplet ranking loss [111] based on the retrieval accuracy on NQ dataset. Different from the above loss functions, **binary cross-entropy (BCE)** loss is more commonly used to optimize the reranker model [188] (detailed in Section 6). It firstly utilizes the dense representations of query and text to derive a match vector, and then predicts the relevance probability of a query-text pair based on the match vector:

$$\Pr(\text{rel} = 1 | q_i, d_i) = \sigma\Big(g(\phi(q_i) \odot \psi(d_i))\Big), \tag{4}$$

where $\sigma(\cdot)$ is the sigmoid function, $g(\cdot)$ is a linear function and $\odot$ is vector combination operation (e.g., concatenation). Then, the BCE loss can be constructed as follows:

$$\mathcal{L}(\langle q_i, d_i \rangle) = -y_{q_i,d_i} \cdot \Pr_{q_i,d_i} - (1 - y_{q_i,d_i}) \cdot (1 - \Pr_{q_i,d_i}),$$

where $d_i$ can be either positive or negative, and $\Pr_{q_i,d_i} = \Pr(\text{rel} = 1 | q_i, d_i)$ defined in Equation (4).

**Similarity measurement.** To instantiate the function $f(\cdot)$, various vector similarity (or distance) measurements can be used to compute the query-text similarity, including inner product, cosine similarity, and Euclidean distance. Several studies have examined the effect of different similarity

functions on the retrieval performance. Thakur et al. [260] conduct an analysis experiment by training two BERT-based models (an identical parameter configuration on MS MARCO dataset) with cosine similarity and inner product, respectively. They observe that the variants with cosine similarity and inner product prefer retrieving shorter and longer documents, respectively. Karpukhin et al. [111] also examine the effect of Euclidean distance as the distance function, however, no significant differences were observed in retrieval performance. So far, in the literature, inner product has been widely adopted as the similarity measurement.

*4.1.2 Incorporating Optimization Constraints.* Besides the above formulation, there are several variants that aim to improve the optimization objective by incorporating more constraints.

**Text-oriented optimization.** In the above, the negative log-likelihood loss (Equation (3)) is modeled in a query-oriented way, which optimizes the likelihood of texts conditioned on the query and the text set. Similarly, we can model the relevance in a text-oriented way as follows:

$$\mathcal{L}_T(\langle q_i, d_i^+ \rangle) = -\log \frac{e^{f(\psi(d_i^+), \phi(q_i))}}{e^{f(\psi(d_i^+), \phi(q_i))} + \sum_{q^- \in Q^-} e^{f(\psi(d_i^+), \phi(q^-))}}, \tag{5}$$

where a set of sampled negative queries denoted by $Q^-$ is incorporated to compute the negative query likelihood. Given both query-oriented and text-oriented loss functions, it has become an important optimization trick to train the dense retrievers by jointly optimizing Equations (3) and (5). Similar symmetric optimization tricks have been used in a number of follow-up studies [132, 134, 290, 295], showing an improved retrieval performance.

**Text-text similarity constraints.** Besides directly optimizing query-text similarities, Ren et al. [218] find that it is useful to incorporate *text-text* similarity constraints in the optimization objective. They argue that previous optimization goals are mainly query-centric, which is difficult to discriminate between positive texts and semantically similar negative texts. Therefore, a text similarity constraint has been proposed, assuming that the similarity between positive passage $d^+$ and query $q$ should be larger than the similarity between positive passage $d^+$ and negative passage $d^-$, i.e., $f_{\text{sim}}(d^+, q) > f_{\text{sim}}(d^+, d^-)$. Such a loss is formulated as follows:

$$\mathcal{L}_{TT}(\langle q_i, d_i^+ \rangle) = -\log \frac{e^{f(\psi(d_i^+), \phi(q_i))}}{e^{f(\psi(d_i^+), \phi(q_i))} + \underline{\sum_{d' \in \mathcal{N}_{q_i}} e^{f(\psi(d_i^+), \psi(d'))}}}. \tag{6}$$

The major difference between Equations (3) and (6) lies in the underlined part, where it incorporates text-text similarity as the normalization term. Not only optimizing the similarity between query and positive text, this approach tries to increase the distance between positive and semantically similar negative texts.

*4.1.3 Challenges for Training Dual-encoders.* Despite the conceptual simplicity of bi-encoders, it is nontrivial to train capable bi-encoders for dense retrieval (Equation (3)). There are three major training issues for the effective training of bi-encoders.

• Issue 1: *Large-scale candidate space.* In retrieval tasks, there is typically a huge number of candidates in a large text collection, while only a few texts among them are actually relevant to a query. Consequently, it is challenging to train capable dense retrievers that perform well on large-scale text collections. Specifically, due to the computational limits, we can sample only a small number of negative samples for computing the loss function (Equation (3)) during training, resulting in a shift in the candidate space between training (a small population of texts) and testing (the entire collection). It has been widely found the sampled negatives have a significant effect on the retrieval performance [111, 207, 289].

• Issue 2: *Limited relevance judgements*. In practice, it is difficult to construct large-scale labeled data with complete relevance judgements for dense retrieval. Even though several benchmark datasets with large sizes have been released (e.g., MS MARCO), it is still limited to training very large PLM-based dense retrievers. Additionally, the relevance annotations in these datasets are far from complete. Therefore, "*false negatives*" (actually relevant but not annotated) are likely to occur in negative sampling, which has become a major challenge for training dense retrievers [207].

• Issue 3: *Pretraining discrepancy*. PLMs are typically trained using pre-designed self-supervised loss functions, such as masked word prediction and next sentence prediction [53]. These pretraining tasks are not specially optimized for the retrieval tasks [28, 165, 213], thus likely leading to suboptimal retrieval performance. Besides, to represent a text, existing work usually adopts the "[CLS]" embedding of PLMs as the text representation, while the "[CLS]" embedding is not explicitly designed to capture the semantics of the whole text [65]. Considering this issue, it needs to design specific pretraining tasks that are more suited to dense retrieval for the underlying PLMs.

The above three issues have become the major technical bottlenecks for dense retrieval, attracting much research attention. Next, we will review the recent progress on addressing the above issues, and discuss three major techniques for dense retrieval, detailed in the following three subsections: Section 4.2 (negative selection for tackling Issue 1), Section 4.3 (auxiliary and synthetic data enrichment for tackling Issue 2), Section 4.4 (knowledge distillation for tackling Issue 2), and Section 4.5 (pretraining for tackling Issues 2 and 3).

### 4.2 Negative Selection

To optimize the dense retriever (Equation (3)), a certain number of sampled negatives are needed for computing the negative log-likelihood loss. Thus, how to select high-quality negatives has become an important issue for improving the retrieval performance. Next, we will review three major negative selection methods, and then present the theoretical discussions.

*4.2.1 In-batch Negatives.* A straightforward approach for negative selection is random sampling, i.e., each positive text is paired with several random negatives. However, most PLMs are optimized in a batch mode on GPU with limited memory, which makes it infeasible to use a large number of negatives during training. Considering this problem, in-batch negatives are used for optimizing the dense retriever: given a query, the positive texts paired with the rest queries from the same batch are considered as negatives. Assume that there are $b$ queries ($b > 1$) in a batch and each query is associated with one relevant text. With in-batch negative technique, we can obtain $b - 1$ negatives for each query in the same batch, which largely increases the number of available negatives per query under the memory limit. The in-batch technique was firstly proposed in Reference [85] for the response selection task of *Smart Reply*, while it has been explicitly utilized for dense retrieval by DPR [111]. In-batch negatives are shown to be effective to improve the learning of bi-encoder by increasing the number of negatives [111, 207].

*4.2.2 Cross-batch Negatives.* By reusing the examples from a batch, in-batch negative training can increase the number of negatives for each query in a memory-efficient way. To further optimize the training process with more negatives, another improvement strategy called *cross-batch negatives* [70, 207] are proposed under the multi-GPU setting. The basic idea is to reuse examples across different GPUs. Assume there are $a$ GPUs for training the dense retriever. We first compute the text embeddings at each single GPU, and then communicate them across all the GPUs. In this way, the text embeddings from other GPUs can be also used as negatives. For a given query, we obtain $a \times b - 1$ negatives from $a$ GPUs, which is approximately $a$ times as in-batch negatives. In this way, more negatives can be used during training for improving the retrieval performance. The idea of cross-batch negatives can be also extended to a single-GPU setting with the technique of

gradient caching [70], where one can accumulate multiple mini-batches for increasing the number of negatives (while taking a larger computational cost).

*4.2.3 Hard Negatives.* Although in-batch and cross-batch negatives can increase the number of available negatives, they cannot guarantee to generate *hard negatives*, which refer to the irrelevant texts but having a high semantic similarity with the query. It is particularly important to incorporate hard negatives to improve the capacity in discriminating between relevant and irrelevant texts [111, 207, 289]. A number of studies have designed different hard negative selection strategies for improving the retrieval performance. According to whether the negative selector (i.e., a sampler over the text collection based on some relevance model) is fixed or updated, hard negatives can be roughly divided into *static hard negatives* and *dynamic hard negatives* [289, 308]. Besides, the selected hard negatives might contain noisy data (e.g., false negatives), and thus *denoised hard negatives* are also used to train dense retrievers [207]. Next, we present the detailed discussions for each kind of hard negatives.

**Static hard negatives.** For static hard negatives, the negative selector is fixed during the training of the dense retriever. The goal is to select negatives that are difficult to be discriminated by the dense retriever. To achieve this, a straightforward way is to sample negatives from top retrieval results from some other retriever, either sparse or dense. Since BM25 [178] usually gives very competitive retrieval performance, several studies select hard negatives based on BM25, i.e., sampling lexically similar texts (but without containing the answers) returned by BM25 [111]. In Reference [156], multiple kinds of negatives are also mixed for training, including retrieved negatives (based on BM25, coarse and fine semantic similarity) and heuristics-based context negatives. This study shows that an ensemble approach combining models trained with mixed hard negatives is able to improve the retrieval performance. As a follow-up study, the authors [157] further design three fusion strategies to combine different kinds of negatives, namely, mixing fusion, embedding fusion and rank fusion. Besides, Hofstatter et al. [92] propose a topic-aware sampling method to compose training batches for dense retrieval, which first clusters the queries before training and then samples queries out of one cluster per batch. In this way, the examples in a batch are highly similar, which implicitly derives hard negatives with the in-batch sampling technique.

**Dynamic (or periodically updated) hard negatives.** As discussed above, static hard negatives are obtained from a *fixed* negative selector. Since the training of dense retriever is an iterative process, it is better to use adaptively updated negatives, called *dynamic hard negatives*, for model training. The ANCE approach [289] proposes to sample from the top retrieved texts by the optimized retriever itself as negatives, which they call *global hard negatives*. It has been shown in Reference [289] that globally selected negatives can lead to faster learning convergence. For retrieving global negatives, it needs to refresh the indexed text embeddings after updating the model parameters, which is very time-consuming. To reduce the time cost, ANCE uses an asynchronous index refresh strategy during the training, i.e., it performs a periodic update for each $m$ batches. Furthermore, Zhan et al. [308] propose a new approach *ADORE* for selecting dynamic hard negatives, which samples negatives from dynamic retrieval results according to the being-updated query encoder. Unlike ANCE [289], ADORE fixes the text encoder and the text embedding index, and utilizes an adaptive query encoder to retrieve top-ranked texts as hard negatives. At each iteration, since the query encoder for negative selection is optimized during training, it can generate adaptive negatives for the same queries. A note is that before training with dynamic hard negatives, the model should be warmed up (i.e., BM25 and the STAR training approach in Reference [289]). To better understand the effect of static and dynamic negatives, we can roughly take an adversarial perspective to illustrate their difference: for a given query, static approaches use fixed negatives during

training (i.e., fixed generator), while dynamic approaches generate adaptive negatives according to the being-optimized retriever (i.e., adaptive generator). Intuitively, dynamic hard negatives are more informative to train the dense retrievers (i.e., the discriminator). Besides, considering the large-scale text corpus, dynamic negatives can potentially alleviate the training-test discrepancy in the candidate space, since it can "*see*" more informative negatives during training.

**Denoised hard negatives.** Negatives play a key role in the training of dense retriever, and it has been found that dense retrievers are actually sensitive to the quality of negatives [202, 207]. When the sampled texts contain noisy negatives, they tend to affect the retrieval performance. This issue becomes more severe for hard negatives, which are more likely to be false negatives [4, 23, 324] (i.e., unlabeled positives), because they are top-ranking retrieval results of high relevance scores to queries. To resolve this issue, RocketQA [207] proposes a denoised negative selection approach, and it utilizes a well-trained cross-encoder to filter top-ranked texts that are likely to be false negatives. Since the cross-encoder architecture is more powerful in capturing rich semantic interactions, it can be utilized to refine the selected hard negatives from the bi-encoder. Given the top-ranked texts retrieved by a bi-encoder, only the predicted negatives with confident scores by the cross-encoder are retained as final hard negatives. In this way, the originally selected negatives are denoised, which are more reliable to be used for training. More recently, SimANS [323] propose to sample *ambiguous negatives* that are ranked near the positives, with a moderate similarity (neither too hard nor too easy) to the query. They empirically show that such negatives are more informative, and less likely to be false negatives.

*4.2.4 Discussions on the Effect of Negative Sampling.* As discussed in previous subsections, a number of negative sampling approaches have been developed to enhance the retrieval performance. Here, we present some discussions about the effect of negative sampling for dense retrieval.

It has been shown that in-batch sampling (Section 4.2.1) cannot generate sufficiently informative negatives for dense retrieval [156, 157, 289]. In particular, Lu et al. [157] analyze why in-batch negatives may not include informative negatives. They cast the negative log-likelihood objective with in-batch negatives as a special case of the ranking-based **Noise Contrastive Estimation (NCE)**. Instead of sampling a negative from the whole collection, in-batch sampling considers a rather small collection (reflecting a different distribution) for sampling, i.e., the annotated relevant texts to the queries in the query set. A similar discussion about the informativeness of in-batch negatives is also presented in Reference [289]: Since the batch size and the number of informative negatives are significantly smaller than the collection size, the probability of sampling informative negatives from a random batch (or mini-batch) tends to be close to zero. Besides, the study in Reference [289] shows that the informativeness of negatives is key to the training of dense retrieval models, from the perspective of convergence rate w.r.t. gradient norms. Furthermore, Zhan et al. [308] show that random negative sampling and hard negative sampling indeed optimize different retrieval objectives: random negative sampling mainly minimizes the total pairwise errors, which might over-emphasize the optimization of difficult queries; while in contrast, hard negative sampling minimizes top pairwise errors, which is more suited for optimizing top rankings.

### 4.3 Auxiliary and Synthetic Data Enrichment

To train PLM-based dense retrievers, the amount of available relevance judgements is usually limited w.r.t. the huge number of model parameters. Therefore, it is important to increase the availability of (pseudo)relevance judgement data. Next, we discuss two approaches for enriching the labeled data, namely, collecting auxiliary labeled data and generating synthetic labeled data.

*4.3.1 Auxiliary Labeled Data Collection.* Several studies propose to incorporate auxiliary labeled datasets for enriching the relevance judgements. Karpukhin et al. [111] collect five datasets of NQ [116], TriviaQA [109], WebQuestions [13], TREC [11], and SQuAD [211], then train a multi-dataset encoder by leveraging all the training data (excluding the SQuAD dataset), and test the unified encoder on each of the five datasets. As shown in Reference [111], the performance on most of the datasets benefits from more training examples, especially the smallest dataset in these five datasets. They further conduct an analysis experiment by training DPR on NQ dataset only and testing it on the WebQuestions and CuratedTREC datasets. The results show that DPR transfers well across different datasets (focusing on the QA task), with some slight performance loss. Furthermore, Maillard et al. [173] propose to train a universal dense retriever across multiple retrieval tasks. Specifically, they devise two simple variants (Variant 1: separate query encoders and shared passage encoders for multiple tasks; Variant 2: shared query and passage encoders for multiple tasks) of DPR, and examine the universal retrieval capacity with multi-dataset training. They show that such a multi-task trained model can yield comparable performance with task-specific models and achieves a better performance in a few-shot setting. Besides leveraging multiple QA datasets, one can also utilize different types of data sources. Oguz et al. [192] propose a unified open-domain question answering approach by utilizing multiple resources of text, tables, lists, and knowledge bases. The basic idea is to flatten the structured data into plain texts, so that we can process these data in a unified text form. As shown in Reference [192], overall, it is useful to combine multiple sources in the experiments on five QA datasets, in both per-dataset and multi-dataset settings. When there is no training data for the target task, it becomes the *zero-shot retrieval*. In this setting, though we can leverage auxiliary datasets for allievating the data scarcity, it should be noted that the final performance is highly affected by these auxiliary datasets [220, 241, 310] (detailed in Section 7.1).

*4.3.2 Synthetic Labeled Data Generation.* In addition to existing datasets, several studies propose to directly generate pseudo question-text pairs for retrieval tasks. Specially, these data generation approaches can be divided into two major categories, either a pipeline or end-to-end way.

For the first category, Alberti et al. [2] propose a pipeline approach to constructing question-answer pairs from large text corpora. It consists of three major stages, including answer extraction, question generation and roundtrip filtering. Experimental results show that pretraining on the synthetic data significantly improves the performance of question answering on SQuAD 2.0 and NQ datasets. In a similar way, Lewis et al. [123] further introduce passage selection and global filtering to construct a dataset called *PAQ* for question answering, which contains 65 million synthetically generated question-answer pairs from Wikipedia. Based on the PAQ dataset, a related study [193] pretrains the bi-encoder retrievers, leading to consistent performance improvement over the variants pretrained with the tasks of ICT and BFS (Section 4.5.1).

Furthermore, several studies [135, 163, 214, 277] also explore the generation-augmented training approach in the *zero-shot setting*, where there are no training datasets in the target domain. Their results show that synthetic data generation is useful to improve the zero-shot retrieval capacity of dense retrievers. We will discuss more on zero-shot retrieval in Section 7.1.

### 4.4 Knowledge Distillation

Considering that relevance judgement is limited, knowledge distillation becomes an important approach to improving the capacity of the bi-encoder. In machine learning, knowledge distillation refers to the process of transferring knowledge from a more capable model (called *teacher*) to a less capable model (called *student*) [87]. Following this convention, our goal is to improve the standard bi-encoder (the student) with a more powerful model (the teacher) on a given labeled dataset.

**Training the teacher network.** To implement the teacher network, we can adopt a well-trained cross-encoder for knowledge distillation, since it is more capable in modeling fined-grained semantic interaction across queries and texts. Typically, the training of the teacher network is independent from the training of the student network. As an improved approach, RocketQA [207] introduces an improved strategy by incorporating the information interaction between the teacher and student when training the teacher network. The basic idea is to randomly sample the top-ranked texts from the student network as negatives for training the teacher. This strategy enables the cross-encoder to adjust to the retrieval results by the bi-encoder. Besides, dual teachers, respectively trained with pairwise and in-batch negatives, are adopted to improve the retrieval performance of the student network [92]. Further, an empirical study [89] is conducted to analyze the effectiveness of knowledge distillation with a single teacher and a teacher ensemble, and it has been found that the performance improves with increasing effectiveness of a single teacher or the ensemble of teachers. While, it should be noted that the teacher network is not necessary to be the cross-encoder. In principle, any retriever that is more capable than the basic student network can serve as the teacher network. For example, Lin et al. [141] explore the possibility of using an enhanced bi-encoder (i.e., ColBERT [113] that uses late interaction) as the teacher network. As another interesting work, Yu et al. [302] utilize a well-trained ad hoc dense retriever to improve the query encoder in a conversational dense retriever, to mimic the corresponding ad hoc query representation.

**Distillation for the student network.** After training the teacher network, we utilize it to improve the student network. The basic idea is to run the well-trained teacher network on the unlabeled (or even labeled) data to produce pseudo relevance labels for training the student network. According to the types of the derived labels, we can categorize the distillation approaches into two major categories, namely, hard-label and soft-label distillation.

*Hard-label distillation.* Given the unlabeled texts, the first approach directly sets the binary relevance labels for unlabeled texts according to the relevance scores of the teacher network. Since the predictions of the teacher network might contain noise or errors, thresholding methods are often adopted to remove texts with low confidence scores. For example, RocketQA [207] generates pseudo relevance labels for top-ranked passages according to their ranking scores: positive (higher than 0.9) and negative (lower than 0.1), and discards the rest with unconfident predictions. They also manually examine a small number of denoised texts, and find that this method is generally effective to remove false negatives or positives.

*Soft-label distillation.* Instead of using hard labels, we can also approximate the outputs of the teacher network by tuning the student network. Formally, let $r^{(t)}_{q,d}$ and $r^{(s)}_{q,d}$ denote the relevance scores of text $d$ w.r.t. to query $q$ assigned by the teacher network and the student network, respectively. Next, we introduce several distillation functions.

— MSE loss. This function directly minimizes the difference between the relevance scores between the teacher and student using mean-squared error loss:

$$\mathcal{L}^{KD}_{MSE} = \frac{1}{2} \sum_{q \in \mathcal{Q}} \sum_{d \in \mathcal{D}} \left( r^{(t)}_{q,d} - r^{(s)}_{q,d} \right)^2, \tag{7}$$

where $\mathcal{Q}$ and $\mathcal{D}$ denote the sets of queries and texts, respectively.

— KL-divergence loss. This function first normalizes the relevance scores of candidate documents into probability distributions by queries, denoted by $\tilde{r}^{(t)}_{q,d}$ and $\tilde{r}^{(s)}_{q,d}$, respectively, and then reduces their KL-divergence loss:

$$\mathcal{L}^{KD}_{KL} = - \sum_{q \in \mathcal{Q}, d \in \mathcal{D}} \tilde{r}^{(s)}_{q,d} \cdot \left( \log \tilde{r}^{(s)}_{q,d} - \log \tilde{r}^{(t)}_{q,d} \right). \tag{8}$$

— Max-margin loss. This function adopts a max-margin loss for penalizing the inversions in the ranking generated by the retriever:

$$\mathcal{L}^{KD}_{MM} = \sum_{q,d_1,d_2} \max\left(0, \gamma - \text{sign}\left(\Delta^{(t)}_{q,d_1,d_2} \times \Delta^{(s)}_{q,d_1,d_2}\right)\right), \tag{9}$$

where $q \in \mathcal{Q}$, $d_1, d_2 \in \mathcal{D}$, $\Delta^{(t)}_{q,d_1,d_2} = r^{(t)}_{q,d_1} - r^{(t)}_{q,d_2}$, $\Delta^{(s)}_{q,d_1,d_2} = r^{(s)}_{q,d_1} - r^{(s)}_{q,d_2}$, $\gamma$ is the margin and $\text{sign}(\cdot)$ is the sign function indicating whether the value is positive, negative or zero.

— Margin-MSE loss. This function reduces the margin difference for a positive-negative pair between the teacher and student, via the MSE loss:

$$\mathcal{L}^{KD}_{M-MSE} = \text{MSE}\left(r^{(t)}_{q,d^+} - r^{(t)}_{q,d^-}, r^{(s)}_{q,d^+} - r^{(s)}_{q,d^-}\right). \tag{10}$$

To examine the effectiveness of different distillation functions, Izacard et al. [102] perform an empirical comparison of the above loss functions, and they find that the KL-divergence loss leads to a better distillation performance than MSE loss and max-margin loss for the task of question answering. Further, the researchers empirically find that the Margin-MSE loss is more effective than the other options [89, 92], e.g., pointwise MSE loss.

**Advanced distillation methods.** Recent studies [223, 321] show that knowledge distillation might become less effective when there exists a large capacity gap between the teacher and student models. Thus, instead of using direct distillation, a progressive distillation approach [142, 160, 303] should be adopted when using a strong teacher model, which can be implemented in two ways. As the first way, we use gradually enhanced teacher models at different stages of distillation, to fit the learning of the student model. PROD [142] proposes to use a progressive chain of the teacher model with improved model architectures and increased network layers: 12-layer bi-encoder → 12-layer cross-encoder → 24-layer cross-encoder (given the 6-layer bi-encoder as the student model). ERNIE-Search [160] introduces two distillation mechanisms for reducing the large capacity gap between the teacher and student models, namely, (i) *late-interaction models* (e.g., ColBERT) to *bi-encoder*, and (ii) *cross-encoder* to *late-interaction models* and then to *bi-encoder*. As the second way, we fix the strong teacher model, and gradually increase the difficulty of the distilled knowledge. CL-DRD [303] proposes a curriculum learning approach, and schedules the distillation process with gradually increased difficulty levels: more difficult samples (with a larger query-text similarity) are arranged at a later stage. In the above, we assume that the teacher model is fixed during a distillation process. RocketQA-v2 [219] extends the distillation way by introducing a *dynamic listwise distillation* mechanism: both the retriever (student) and the reranker (teacher) are mutually updated and improved. Based on the KL-divergence loss (Equation (8)), the teacher model is also able to be adjusted according to the student model, yielding a better distillation performance.

### 4.5 Pretraining for Dense Retrieval Models

The original purpose of PLMs is to learn universal semantic representations that can generalize across different tasks. However, such representations often lead to suboptimal performance on downstream applications due to the lack of task-specific optimization [28, 120, 165, 213]. Considering this issue, recent studies employ task-related pretraining strategies for dense retrieval. Besides reducing the pretraining discrepancy (Issue 2), these approaches can also alleviate the scarcity of labeled relevance data (Issue 3). Next, we describe the pretraining approaches for dense retrieval.

*4.5.1 Task Adaptive Pretraining.* This line of pretraining tasks essentially follow what have been used in BERT [53], but try to mimic the retrieval task in a self-supervised way. Below, we list several representative pretraining tasks for dense retrieval.

• ***Inverse Cloze Task (ICT)*** [120] randomly selects a sentence of a given text as the query, and the rest sentences are considered as gold matching text. This task aims to capture the semantic context of a sentence and enhance the matching capacity between query and relevant contexts.

• ***Body First Selection (BFS)*** [28] utilizes the sentences from the first section of a Wikipedia article as anchors to the rest sections of texts. It considers a random sentence from the first section as the query and a randomly sampled passage in the following sections as a matched text.

• ***Wiki Link Prediction (WLP)*** [28] utilizes the hyperlink to associate the *query* with *text*, where a sampled sentence of the first section from a Wikipedia page is considered as the query and a passage from another article containing the hyperlink link to the query page is considered text.

• ***Recurring Span Retrieval (RSR)*** [213] proposes to use *recurring spans* (i.e., ngrams with multiple occurrences in the corpus) to associate related passages and conduct unsupervised pretraining for dense retrieval. Given a recurring span, it first collects a set of passages that contain the recurring span, and then transforms one of the passages into a query and treats the rest passages as positives.

• ***Representative wOrds Prediction (ROP)*** [165] utilizes a document language model to sample a pair of word sets for a given text. Then, a word set with a higher likelihood is considered to be more "representative" for the document, and the PLM is pretrained to predict the pairwise preference between the two sets of words. Following ROP, another variant called *B-ROP* [166] replaces the unigram language model by a BERT model, and samples the representative words from a *contrastive term distribution* constructed based on document-specific and random term distributions.

Instead of using a pipeline way, Shakeri et al. [239] propose an end-to-end approach to generating question-answer pairs based on machine reading comprehension by using a pretrained LM (e.g., BART [121]). It aims to train a sequence-to-sequence network for generating a pair of question and answer conditioned on the input text. Reddy et al. [214] further extend this approach by incorporating data selection for enhancing the generated question-text pairs for retrieval tasks.

*4.5.2 Retrieval-augmented Pretraining.* To enhance the modeling capacity of PLMs, another line of pretraining tasks is to incorporate an external retriever by enriching the relevant contexts. The basic idea is to enhance the training of the **masked language modeling (MLM)** task with more referring contexts from a knowledge retriever.

As a representative work, REALM [79] utilizes a knowledge retriever for retrieving relevant texts from a large corpus, and the retrieved texts are further encoded and attended to augment language model pretraining. In REALM, the basic idea is to reward or discourage the retriever according to whether the retrieved context is useful to improve the prediction of the masksed words. Without using explicit human annotation, the context retriever is further trained via language modeling pretraining, in which the retrieved texts are modeled by a latent variable through marginal likelihood. By fine-tuning the model, REALM performs well on the task of open-domain question answering. It further proposes to use an asynchronous optimization approach based on the maximum inner product search. As an extension work, Balachandran et al. [10] perform a thorough tuning of REALM on a variety of QA tasks. They conduct extensive experiments with multiple training tricks, including using exact vector similarity search, training with a larger batch, retrieving more documents for the reader, and incorporating human annotations for evidence passages. Their experiments show that REALM was not sufficiently trained for fine-tuning, and it is important to design suitable training and supervision strategies for improving open-domain QA systems.

*4.5.3 Representation Enhanced Pretraining.* For bi-encoder based dense retrievers, a typical approach is to utilize the inserted "[CLS]" token to obtain the representation of a query or text.

However, the original "[CLS]" embedding is not explicitly designed to represent the meaning of a whole text. Hence, this may lead to suboptimal retrieval performance, and it needs more effective approaches to enhance the "[CLS]" embedding for dense retrieval.

**Autoencoder enhanced pretraining.** Inspired by the success of autoencoders in data representation learning [49, 128, 247, 248], several studies explore autoencoders to enhance the text representation for dense retrieval. The basic idea is to compress the text information into the "[CLS]" embedding by using an encoder network, and then use a paired decoder to reconstruct the original text based on the "[CLS]" embedding. In this way, the learned "[CLS]" embedding is enforced to capture more sufficient text semantics than the original attention mechanisms in PLMs. Gao et al. [65] propose the Condenser architecture consisting of three orderly parts: early encoder backbone layers, late encoder backbone layers and Condenser head layers (only used during pretraining). The key point is that Condenser removes fully connected attention across late and Condenser head layers, while keeping a short circuit from early layers to Condenser head layers. Since Condenser head layers can only receive information of late backbone layers via the late "[CLS]," the late "[CLS]" embedding is therefore enforced to aggregate the information of the whole text as possible, for recovering the masked tokens. Similar attempts have been made in **Transformer-based Sequential Denoising Auto-encoder (TSDAE)** [276], which is a Transformer-based sequential denoising autoencoder for enhancing the text representation. Furthermore, co-Condenser [66] extends the Condenser architecture by incorporating a query-agnostic contrastive loss based on the retrieval corpus, which pulls close the text segments from the same document while pushing away other unrelated segments. Following the information bottleneck principle [262],[4] recent studies [240, 279] refer to the key representations (e.g., the "[CLS]" embedding) that aims to capture all the important semantics in the above autoencoder approaches as *representation bottlenecks*.

**Unbalanced autoencoder based pretraining.** Recently, Lu et al. [159] have reported an interesting finding that a stronger decoder may lead to worse sequence representations for the autoencoder. It is explained as the *bypass effect*: when the decoder is strong, it may not refer to the information representation from the encoder, but instead perform the sequence generation conditioned on previous tokens. They provide theoretical analysis on this finding, and implement a weak decoder based on a shallow Transformer with restricted access to previous context. In this way, it explicitly enhances the dependency of the decoder on the encoder. This work has inspired several studies that use the "*strong encoder, simple decoder*" architecture for unsupervised text representation pretraining. Based on such an unbalanced architecture, SimLM [279] pretrains the encoder and decoder with replaced language modeling, where it aims to recover the original tokens after replacement. To optimize the dense retriever, it further employs a multi-stage training procedure, which uses hard negative training and cross-encoder distillation. Furthermore, RetroMAE [154] proposes to use a large masking ratio for the decoder (50∼90%) while a common masking ratio for the encoder (15%). It also introduces an enhanced decoding mechanism with two-stream attention and position-specific attention mask. Based on RetroMAE, Liu et al. [143] present a rigorous two-stage pretraining approach (i.e., general-corpus pretraining and domain-specific continual pretraining), which shows strong performance on a variety of benchmark datasets. Besides, LexMAE [240] applies such a representation enhanced pre-training strategy to learned sparse retrieval (e.g., SPLADE [63]) based on PLMs, where it incorporates lexicon-based representation bottleneck (i.e., continuous bag-of-words representations with learned term weights) for pretraining.

[4]In Reference [262], the authors define the goal of information bottleneck as "finding a maximally compressed mapping of the input variable that preserves as much as possible the information on the output variable."

**Contrastive learning-enhanced pretraining.** Another promising direction is to apply contrastive learning (either unsupervised or supervised) to enhance the text representations for dense retrieval. Contrastive learning is originated from the field of computer vision (pioneered by the works of SimCLR [35] and MoCo [84]), where the key idea is to learn the representation of images by making similar ones close and vice versa. Typically, contrastive learning is conducted in two major steps. First, the augmented views are generated for each image using various transformation methods (e.g., cropping and rotation). The augmented views are usually considered as *positives* to the original images, while randomly sampled instances are considered as *negatives* to the original images. Such a data augmentation process is particularly important to the final performance of contrastive learning. Second, a discrimination task is designed assuming that the positives should be closer to the original one than the negatives. Following the same idea, we can generate large-scale positive and negative text pairs for unsupervised pretraining of text embeddings. As a representative study, SimCSE [71] produces positives of one sample text by applying different dropout masks, and uses in-batch negatives. ConSERT [294] uses four ways to generate different views (i.e., positives) for texts, including adversarial attacks, token shuffling, cutoff and dropout. Contriever [101] proposes generating positives by using ICT and cropping (i.e., sampling two independent spans from a text to form a positive pair) and generating negatives by using in-batch and cross-batch texts. Experimental results demonstrate that unsupervised contrastive pretraining leads to good performance in both zero-shot and few-shot settings [101]. In a similar manner, LaPraDoR [290] generates positives by using ICT and dropout, and proposes an iterative contrastive learning approach that trains the query and document encoders in an alternative way. In contrast to autoencoder enhanced pretraining, Ma et al. [164] further propose to pretrain an encoder-only network with a new contrastive span prediction task, which aims to to fully reduce the bypass effect of the decoder. It designs a group-wise contrastive loss, considering the representations of a text and the spans that it contains as positive pairs and cross-text representations as negative pairs. Besides, it has shown that unsupervised contrastive pretraining [184] (pretraining with text pairs and text-code pairs) can also improve both text and code retrieval performance.

**Discussion.** Compared with the pretraining approaches in previous parts (Sections 4.5.1, 4.5.2, and 4.5.3), representation enhanced pretraining does not explicitly use retrieval (or retrieval-like) tasks as optimization goals. Instead, it aims to enhance the informativeness of the text representation, i.e., the "[CLS]" embedding. For dense retrieval, the text representations should capture the semantic meaning of the whole text. By designing effective pretraning strategies (autoencoder or constrastive learning), these approaches can produce more informative text representations, thus improving the retrieval performance [66, 159, 276, 279, 290]. Besides, it has been shown that these approaches can improve the retrieval performance even in zero-shot or low resource settings [65, 101, 290], which also alleviates the scarcity issue of relevance judgements.

## 4.6 Empirical Performance Analysis with RocketQA

Previous sections have discussed various optimization techniques to improve the training of dense retrievers. In this section, we take RocketQA [207] as the studied model and examine how different optimization techniques improve its retrieval performance on benchmark datasets.

*4.6.1 Experimental Setup.* To prepare our experiments, we adopt the widely used MS MARCO passage retrieval dataset [185] for evaluation, and the detailed statistics of this dataset are reported in Table 4. For comparison, we consider two variants of RocketQA, namely, RocketQA [207] and RocketQAv2 [219], and a related extension called RocketQA$_{PAIR}$ (originally called *PAIR*). For RocketQA, it uses three major training tricks, namely, cross-batch negatives (Section 4.2.2, increasing the number of negative examples optimized for each query by reusing data across batches),

Table 3. Empirical Comparison of Different RocketQA Variants on MS MARCO Dataset for the Passage Retrieval Task (in Percentage), Where *bs* Denotes the Batch Size

| Model | Technique | | MRR@10 |
|---|---|---|---|
| BM25 | (0) | BM25 | 18.7 |
| $\text{DPR}_{\text{ERNIE}}$ | (1) | in-batch ($bs$ = 4,096) | 32.4 |
| RocketQA | (2a) | cross-batch ($bs$ = 4,096) | 33.3 |
| | (2b) | cross-batch ($bs$ = 2,048) | 32.9 |
| | (2c) | cross-batch ($bs$ = 1,024) | 31.4 |
| | (2d) | cross-batch ($bs$ = 512) | 29.8 |
| | (2e) | cross-batch ($bs$ = 256) | 29.5 |
| | (2f) | cross-batch ($bs$ = 128) | 28.6 |
| | (2g) | cross-batch ($bs$ = 64) | 26.9 |
| | (2h) | cross-batch + hard neg w/o denoising ($bs$ = 4,096) | 26.0 |
| | (2i) | cross-batch + hard neg w/ denoising ($bs$ = 4,096) | 36.4 |
| | (2j) | cross-batch + denoised hard negatives + data augmentation (208K) ($bs$ = 4,096) | 36.5 |
| | (2k) | cross-batch + denoised hard negatives + data augmentation (416K) ($bs$ = 4,096) | 36.9 |
| | (2l) | cross-batch + denoised hard negatives + data augmentation (832K) ($bs$ = 4,096) | 37.0 |
| $\text{RocketQA}_{\text{PAIR}}$ | (3a) | in-batch + hard-label distillation ($bs$ = 4,096) | 37.2 |
| | (3b) | in-batch + hard-label distillation + p-centric pretraining ($bs$ = 4,096) | 37.9 |
| RocketQAv2 | (4a) | soft-label distillation ($bs$ = 96) | 36.5 |
| | (4b) | soft-label distillation + joint retriever-reranker training (7 hard negs) ($bs$ = 96) | 36.5 |
| | (4c) | soft-label distillation + joint retriever-reranker training (31 hard negs) ($bs$ = 96) | 37.3 |
| | (4d) | soft-label distillation + joint retriever-reranker training (63 hard negs) ($bs$ = 96) | 38.1 |
| | (4e) | soft-label distillation + joint retriever-reranker training (127 hard negs) ($bs$ = 96) | 38.3 |

denoised hard negatives (Section 4.2.3, removing passages from negative passages that are likely positive) and distillation-based data augmentation (Section 4.4, using cross-encoder for distillation to construct more training data). For RocketQAv2, it incorporates a new soft-label distillation mechanism, called dynamic listwise distillation (Section 4.4, jointly training dual-encoder and cross-encoder). For $\text{RocketQA}_{PAIR}$, it is built on RocketQA and incorporates the passage-centric pretraining technique (Section 4.1.1, modeling the relation between query-passage similarity and passage-passage similarity). Considering the tunable configuration (e.g., the number of negatives and batch size) of these models, we include multiple variants with different settings. Although RocketQA variants do not include all the optimization techniques introduced in Section 4, they provide a unified base model by examining the effects of different optimization techniques in a relatively fair way. As a comparison, we incorporate the classic BM25 [227] and DPR [111] methods as baselines. To reproduce the experiments in this part, we implement a code repertory for dense retrieval at the link https://github.com/PaddlePaddle/RocketQA and release the script or code to reproduce the results in Table 3. This code repertory is implemented based on the library PaddlePaddele, consisting of RocketQA, $\text{RocketQA}_{PAIR}$ and RocketQAv2. For a fair comparison, we re-implement the DPR model with PaddlePaddele and use ERNIE as the base PLM [253].

*4.6.2 Results and Analysis.* Table 3 presents the performance comparsion of different methods or variants on the MS MARCO dataset for passage retrieval. We have the following observations:

• First, cross-batch technique is able to improve the retrieval performance. Compared with $\text{DPR}_{ERNIE}$, the RocketQA variants $2a \sim 2g$ have the same configuration and implementation, except the cross-batch technique. Such a technique can lead to significant improvement, i.e., 0.9 percentage point in absolute *MRR*@10 performance (variant 1 versus 2*a*). Besides, it is key to use as more negatives as possible during the cross-batch training, which means that a large batch size should be used.

• Second, it is effective to use denoised hard negatives, which leads to a significant improvement of 3.1 percentage points in absolute *MRR*@10 performance (variant 2*a* versus 2*i*). As a comparison, when using non-denoised hard negatives, the performance decreases quickly. A possible reason is that hard negatives are more likely to be false negatives that will harm the final performance.

• Third, data augmentation can further improve the retrieval performance with both cross-batch technique and denoised hard negatives, which leads to a substantial improvement of 0.6 percentage point in absolute *MRR*@10 performance (variant 2*i* versus 2*l*). Besides, as we can see, the performance improves with the increasing amount of pseudo labeled data.

• Fourth, passage-centric pretraining can boost the performance, leading to a substantial improvement of 0.7 percentage point in absolute *MRR*@10 performance (variant 3*a* versus 3*b*). It characterizes a more comprehensive semantic relation among query, positives, and negatives.

• Finally, the variant with dynamic listwise distillation (variant 4*e*) achieves the best performance among all the RocketQA variants. Dynamic listwise distillation provides a joint training approach for retriever and reranker based on soft-label distillation (detailed in Section 6.2.3).

To conclude the discussion of this part, we can see that effective optimization techniques are key to achieve good retrieval performance for dense retrievers.

## 5 INDEXING FOR DENSE RETRIEVAL

Previously, we have extensively discussed how to design and train a dense retrieval model from an algorithmic perspective. To implement a dense retrieval system, it is important to develop an efficient index structure that can support relevance retrieval in dense vector space. In this section, we discuss the data structures and algorithms for efficient dense retrieval.

### 5.1 Traditional Inverted Index for Sparse Retrieval

To start with, we review the key data structure to support traditional sparse retrieval, i.e., term-based inverted index. In essence, inverted index maintains an efficient mapping from terms to their locations in documents [179]. The basic idea is to construct term-based posting lists by aggregating the occurrences of a term. To be specific, each term in the vocabulary is associated with a unique posting list, and the posting list stores the identifiers of the documents (possibly with more detailed positional information) in which the term occurs. The key point of using the inverted index for relevance evaluation is to examine the documents that at least contain a query term. For retrieval, given a query, it first fetches the postings lists associated with query terms and traversing the postings to compute the result set. As major lexical retrieval toolkits, Apache Lucene and its extensions including Elastic Search and Apache Solr have become *de facto* software for building inverted index-based retrieval systems. Besides, Anserini [296], also built on top of Lucene, allows researchers to easily reproduce sparse retrieval models on standard IR test collections.

### 5.2 Approximate Nearest-neighbor Search for Dense Retrieval

Since dense retrieval does not rely on lexical match, term-based inverted index is no longer suited for embedding-based retrieval. Dense retrieval represents both queries and texts as dense vectors, which can be cast into the problem of *nearest-neighbor search*: finding the most close vector(s) from a collection of candidate vectors (i.e., the texts in the collection) w.r.t. a query vector based on some similarity or distance measurements. In what follows, we first formulate the nearest-neighbor search under the setting of dense retrieval, then discuss two major directions to improve search efficiency, and finally introduce the implementation and software.

*5.2.1 Formulation and Overview.* In this part, we formulate the dense retrieval task as the nearest-neighbor search problem in the vector space of embeddings. Given a collection of

candidate embeddings $\mathcal{P}$ ($\mathcal{P} \subset \mathbb{R}^l$ and $|\mathcal{P}| = m$) and a query embedding $\mathbf{q} \in \mathbb{R}^l$, the goal of nearest neighbor search [15, 242] is to efficiently search for $\mathbf{p} \in \mathcal{P}$ that are most close to $\mathbf{q}$, i.e., $\mathbf{p}^* = \arg\max_{\mathbf{p}\in\mathcal{P}} f_{\text{sim}}(\mathbf{q}, \mathbf{p})$, where $f_{\text{sim}}(\cdot)$ can be implemented by various similarity or distance functions, e.g., inner product or cosine similarity. The most straightforward approach for nearest-neighbor search is to enumerate all the candidate embeddings, i.e., computing the similarity between a query embedding and each candidate embedding in the collection. However, it will be very time-consuming when the number of candidate embeddings is large. To reduce the high computational costs of brute-force enumeration, a number of ***Approximate Nearest-neighbor Search (ANNS)*** algorithms [15, 80, 131, 242] are developed to retrieve the approximates of the exact nearest neighbors, possibly with some loss in search quality.

Generally speaking, the design of ANNS algorithms needs to make a trade-off between search efficiency and quality. To evaluate different ANNS aglorithms, ANN-Benchmarks [8] has been released with a million-scale benchmark, and an enlarged version called Big-ANN-Benchmarks [244] further extends the data size to a billion scale. These two benchmarks maintain public performance rankings of different ANNS algorithms under different settings, which provides a guidance to select suitable algorithms according to dataset size and efficiency/accuracy requirement. In general, there are two feasible directions to improve the search efficiency of ANNS: (1) reducing the amount of similarity computes based on various index structures, and (2) reducing the overhead of each similarity compute by product quantization. In what follows, we will discuss index structures for ANNS (Section 5.2.2) and product quantization algorithms for ANNS (Section 5.2.3), and introduce public ANNS software of for building efficient dense retrieval systems (Section 5.3).

*5.2.2 Improving Search Efficiency by Index Structures.* As a major direction to improve search efficiency, we can design special index structures to reduce the amount of similarity computes, and there are several ways to develop efficient index structure for dense retrieval, including hashing-based approaches, clustering-based inverted index approaches, and graph-based approaches.

• *Hashing-based approaches* [99, 242, 275] assign vectors into different buckets according to a pre-designed hashing function. The idea is to allocate highly similar vectors (i.e., similar texts) into the same buckets. Given a query vector, we employ a hash function to map it into some specific bucket, and then search the vectors only in these buckets. In this way, the search efficiency will be improved. Such an approach is more suitable when the dimension of vectors is small.

• *Clustering-based index approaches* [245] partition the search space by clustering. Given a query vector, we first compare it with the cluster centroids for finding the most similar clusters. Then, we can locate a limited number of clusters that contain the target vectors, and further perform in-cluster search for finding the target vectors. This approach is flexible to use, with high search quality and reasonable efficiency. Further, this approach can be implemented in a memory-disk hybrid way for indexing large-scale data, where cluster centroids are stored in memory while posting lists to centroids are stored in disk [34].

• *Graph-based approaches* [174, 274] (e.g., SPTAG and HNSW) work by navigating a graph that is built by associating the vertices with their nearest neighbors. Given a query, it will search on the graph by greedily selecting the similar neighbors. Such approaches are efficient due to the *small world phenomenon* [114]: the average path length of between two vertices (that might be not directly connected) is short. Graph-based approaches usually perform well on large high-dimensional datasets, while taking a high memory cost.

*5.2.3 Improving Search Efficiency by Product Quantization.* Besides reducing the amount of similarity computes, another possible way is to reduce the overhead of each similarity compute. Different from sparse representations (mostly composed of integer identifiers), dense representations

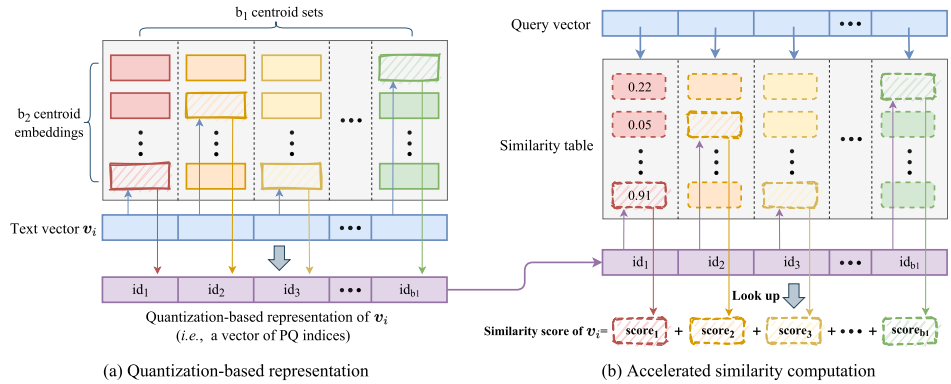



Fig. 3. Illustration of text representation and similarity search based on product quantization (PQ). Here, we assume that there are $b_1$ centroid sets, each with $b_2$ centroid embeddings. In this representation, an original text vector will be mapped to $b_1$ PQ indices, where each PQ index maps to a centroid embedding from the corresponding centroid set. The quantization-based representation of a text is a vector of $b_1$ PQ indices (corresponding to the $b_1$ nearest centroid embeddings). When evaluating the similarity of a text, we can simply sum the entries from the similarity table with its $b_1$ PQ indices.

(i.e., text embeddings) are composed of real-value numbers, and they take significantly more time for similarity computation, leading to an increase in memory cost.

In this part, we describe a ***product quantization* (PQ)** approach [72, 105] to compressing the text embeddings and accelerating similarity compute, reducing both time and memory costs. In practice, product quantization can be also jointly used with efficient index structures, e.g., clustering-based index and graph-based index. In the following, we will first describe how to compress the text embeddings as quantization-based representations, and then introduce how such representations reduce the cost of similarity computation when searching for a query. To introduce product quantization, we focus on discussing the case of a single embedding vector $\boldsymbol{v}_i \in \mathbb{R}^l$, while it can be easily extended to a set of text embeddings.

**Quantization-based representations.** As shown in Figure 3(a), to compress the text embeddings, given a corpus of $m$ texts, product quantization [72, 105] firstly constructs $b_1$ centroid sets of embeddings, with each consisting of $b_2$ centroid embeddings. To represent an $l$-dimensional vector $\boldsymbol{v}_i$, we first split it into $b_1$ equal-length subvectors $\{\boldsymbol{v}_i^{(j)}\}_{i=1}^{b_1}$. Then, the $j$th subvector $\boldsymbol{v}_i^{(j)} \in \mathbb{R}^{l/b_1}$ from the vector $\boldsymbol{v}_i$ is mapped to the nearest centroid (recorded as a *PQ index* ranging from 1 to $b_2$) in the corresponding $j$th centroid set. Since the number of centroid embeddings in a centroid set is usually smaller than 256 (i.e., $b_2 < 256$), we can use one byte to represent a PQ index, which is able to largely reduce the space cost. Finally, the original vector is compressed as a vector of $b_1$ PQ indices (with $b_1$ bytes), called *quantization-based representation*. The quantization-based representations of all text embeddings in a corpus can be pre-computed and stored before search.

**Accelerated similarity computation.** Figure 3(b) illustrates how similarity computation can be implemented in an efficient way by using quantization-based representations. First, a query embedding is split into $b_1$ subvectors in the same way as that text vectors are split. Then, we compute the similarity between each query subvector and the centroid embeddings in the corresponding centroid set. As a result, a similarity table is constructed with $b_1 \cdot b_2$ entries (each entry represents the similarity score between one query subvector and one centroid embedding), and the table entries can be accessed by the PQ indices. To evaluate the similarity of a text w.r.t. the query, we

employ the quantization-based representation of a text (i.e., a vector of $b_1$ PQ indices) to look up the similarity table, and then sum the corresponding $b_1$ entries as the similarity score of the text. Note that the similarity table will be shared for all the texts, i.e., the entire corpus only requires a total of $b_1 \cdot b_2$ embedding similarity computes, independent of the large corpus size. With this similarity table, the similarity evaluation of a text can be efficiently implemented by table lookup.

**Optimizing quantization-based index for retrieval.** The original purpose of product quantization is to solve general vector compression problems, and it cannot be directly end-to-end learned in neural network models, due to the involved non-differentiable operations. Besides, dense retrieval requires specific optimization strategies (e.g., negative sampling), which are not supported by product quantization techniques. To address the above issues, Zhan et al. [306] propose an end-to-end learning approach based on product quantization for dense retrieval, where three major optimization strategies are proposed, including ranking-oriented loss, centroid optimization, and end-to-end negative sampling. Based on Reference [306], the authors further incorporate the technique of constrained clustering to achieve a better centriod assignment for a given document [307]. Besides, Zhang et al. [312] also propose several improved training strategies to jointly optimize deep retrieval models with product quantization-based embedding index. Yamada et al. [292] incorporate a learned hash function to represent both queries and texts as binary codes in an end-to-end way. The proposed model achieves comparable performance to DPR, and further significantly reduces the computational and memory costs.

Besides efficiency improvement, it has been found that discrete representations derived from dense embeddings are useful to interpret the retrieval results by analyzing different aspects of input that a dense retriever focuses on Reference [305].

### 5.3 Implementation for ANNS Algorithms

In dense retrieval systems, efficient similarity search is used at multiple stages. The primary use is to recall the most relevant text embeddings given the query vector. Besides, it can be also applied to select the (static or dynamic) hard negatives or retrieve supporting context. For example, REALM [79] implements a textual knowledge retriever that retrieves related context information for a given query, and ANCE [289] adopts it to identify global negatives based on vector search.

As aforementioned, most of existing dense retrieval studies adopt Faiss [108] for nearest-neighbor search, which is a publicly released library for efficiently searching and clustering large-scale dense vectors. Basically, Faiss provides the exact nearest-neighbor search function, which is widely used by previous studies [207, 289]. Moreover, it supports the implementations of several ANNS approaches discussed in Section 5.2.3 and Section 5.2.2 (including clustering-based approaches, graph-based approaches, and quantization approaches) in both CPU implementations and GPU implementations, since efficiency is a significant factor in practice. It also supports different similarity functions for vector search, including dot product and cosine similarity. To meet the requirements of effectiveness and efficiency, users can set the appropriate configuration with Faiss according to different hardware profile and data sizes. Besides, HNSWlib [175] and SPTAG [33] are libraries focusing on the efficient implementation for graph-based approach. ScaNN [78] is developed based on quantization-based approaches, which achieves excellent performance on ANN-Benchmarks. Distributed-Faiss [200] provides the distributed solutions for ANNS, when the index is too large to fit into the memory of a single machine. This survey mainly limits the discussion to in-memory solutions. When the data size is extremely large (e.g., 100 billion scale), it is generally not feasible to use completely in-memory ANNS index for dense retrieval, where we need to develop multi-level index using a hybrid of memory index and disk index, e.g., DiskANN [250] and SPANN [34].

## 6 INTEGRATION WITH RERANKING

In a complete retrieval system, it usually consists of first-stage retrievers and subsequent rerankers. In this section, we first briefly introduce the retrieval pipeline, and then present several approaches to optimizing the retrieval pipeline.

### 6.1 Retrieval Pipeline

This part introduces the retrieval pipeline and PLM-based rerankers.

*6.1.1 General Retrieval Pipeline.* To start our discussion, we consider a simplified retrieval pipeline, consisting of two major stages, namely, the first-stage retrieval and the reranking. To obtain the final ranked list, a certain number of possibly relevant documents (e.g., several hundreds to thousands) to a given query are retrieved from a corpus by a retriever, such as dense retriever or BM25. Then, the candidate texts are scored and reranked by a more capable relevance model (called *reranker*). Finally, top-ranked texts (e.g., several or tens) will be returned as the search results to downstream tasks, such as question answering and dialog systems. Generally, first-stage retrieval and reranking stages have different focuses in a retrieval system [9, 137, 188]. First-stage retrieval aims to efficiently recall relevant candidates from a large text corpus. As a result, it is practically infeasible to employ time-consuming models in first-stage retrieval. In contrast, the goal of reranking is to reorder the candidate results from the proceeding stages, where the number of candidate texts is generally smaller (e.g., hundreds or thousands) and more complicated models can be used to implement the rerankers. Therefore, bi-encoder is often used for implementing the retriever, and cross-encoder is often used as the architecture of the reranker. Note that a retrieval pipeline usually contains multiple reranking stages by successively refining a reduced candidate list for producing the final results. Besides, there may also exist multiple first-stage retrievers in a practical retrieval system, where the results from multiple retrievers are aggregated as the input of the rerankers. Interested readers can refer to References [9, 57, 137] for detailed discussions.

*6.1.2 PLM-based Rerankers and Multi-stage Ranking.* In this part, we first discuss the typical reranker models based on PLMs, and then introduce multi-stage ranking mechanism that involves PLM-based rerankers.

**Reranker models.** To implement the reranker, a typical approach is to employ the cross-encoder as the ranking model, showing substantial improvements over the traditional methods [188, 204, 284, 293]. Specifically, BERT is widely used to implement the cross-encoder for estimating the relevance score [188, 204], e.g., monoBERT [191]. As the input, a query and a candidate text are concatenated as a sequence proceeded by the "[CLS]" token. After encoding the query-text sequence, the "[CLS]" embedding is adopted as the match representation between the query and text, which will be subsequently fed into a neural network for computing the relevance score of the text being relevant (Equation (4)). The relevance score is computed for each text independently, and the final ranking of texts is obtained by sorting them according to relevance scores. For training the rerankers, it usually formulates the ranking problem as a binary classification task [188] using the BCE loss as shown in Equation (5). To optimize the BCE loss, it also needs to generate negatives for learning, which can be randomly selected or sampled from the top results of the retriever [68, 325]. Furthermore, duoBERT [191] implements the BERT-based text ranking in a pairwise way, where it takes as input a query and a pair of texts to compare (with a concatenation pattern "[CLS] *query* [SEP] $text_1$ [SEP] $text_2$"). The training objective is to reserve the partial order of semantic relevance for a given text pair, such that it can predict the relevance preference for ranking the texts. Besides encoder-only models, encoder-decoder-based PLMs (e.g., T5) have

been also utilized to implement the reranker, which takes as input a query-text pair and outputs the relevance label [189] or ranking score [330].

**Multi-stage ranking.** To construct a complete system, a commonly used strategy is to arrange the retriever and the reranker(s) in a processing pipeline. Nogueira et al. [191] present a multi-stage ranking architecture to construct an effective retrieval pipeline. In this architecture, the first-stage ranker is implemented by the BM25 algorithm, while the second-stage and third-stage rankers are implemented by the monoBERT and duoBERT, respectively. In this approach, different stages score each candidate separately. Recently, a new retriever-reranker integration architecture has been proposed [318] for multi-stage ranking optimization (a similar approach SetRank [194] has been proposed). As the major novelty, the learned relevance information from retriever and reranker are transformed into input embeddings and positional embeddings, respectively, for another Transformer encoder, which produces the final ranking scores. Instead of scoring each candidate individually, the Transformer encoder derives the ranking scores by taking a listwise contrastive loss. Thus, the relevance information from the retrieval and reranking stages can be effectively utilized and optimized in a listwise way. In practice, multi-stage ranking is widely adopted in production systems (e.g., Facebook Search [95] and Walmart Product Search [171]), where more comprehensive factors are considered and more complicated strategies are designed.

Besides, there is a large body of studies that utilize PLMs to enhance pre-BERT neural ranking models, e.g., CEDR [170]. In these studies, PLMs are used to generate contextual features to enhance the semantic representation or similarity, which do not serve as the ranking backbone. Thus, we omit the discussion of this topic, and interested readers can find a detailed discussion in References [22, 28, 168].

### 6.2 Retrieval Pipeline Training

The retrieval pipeline forms a cascaded structure by stacking the retriever and (one or multiple) reranker(s), and it needs to design suitable optimization algorithms for the entire pipeline. Next, we introduce three major optimization approaches for the retrieval pipeline. For simplicity, we only discuss the scenario where only a retriever and a reranker are involved in a retrieval pipeline.

*6.2.1 Separate Training.* A straightforward approach is to separately optimize the retriever and reranker, considering them as two independent components. In this approach, the training of the retriever and reranker are transparent to each other without information interaction (except the retrieval results). Typically, we can firstly optimize the retriever and then learn the reranker. Such an optimization approach cannot sufficiently capture the cascading information correlations between different stages in the retrieval pipeline, thus possibly leading to a suboptimal ranking performance. The reasons are twofold. First, when the retrieved candidate list improves, the contained negatives also become increasingly difficult to be discriminated by the reranker, which are more likely to be false negatives [68, 207]. Second, without considering the first-stage retrieval results, the optimization of reranker cannot be well adjusted to the result distribution of the retriever, which is not specially optimized according to the entire pipeline [219]. Therefore, it is necessary to incorporate information interaction between the retriever and reranker during training.

*6.2.2 Adaptive Training.* As an improvement approach, adaptive training enables the information interaction between the retriever and reranker. The basic idea is to let the two components adapt to the retrieval results (or intermediate data representations) of each other to achieve a better overall retrieval performance. For example, Qu et al. [207] propose to train a cross-encoder by sampling from top results of the retriever (a dual-encoder) as negatives, which lets the cross-encoder adjust to the distribution of the retrieval results by the dual-encoder. Another commonly

adopted way is to alternatively update the retriever and reranker during the training process. At each iteration, the fixed component can provide necessary relevance information or guidance signals to another being-optimized component. Trans-encoder [144] designs an iterative joint training framework by alternating between the cross-encoder and bi-encoder. Specifically, during the learning process, one component will produce pseudo relevance labels for updating the other one. Furthermore, an adversarial learning approach has been proposed in Reference [311]: the roles of the retriever and ranker are to retrieve negative documents and identify ground-truth documents from a mixed list including both positive and negative ones, respectively. The entire optimization process of the retriever and ranker is formulated as a minimax game.

*6.2.3 Joint Training.* In implementation, retriever and reranker are often optimized in different ways, which makes it difficult for joint optimization. Specially, the retriever is usually trained by maximizing the probabilities of positive passages against a list of sampled negatives. It is optimized according to the overall ranking for the candidate list of positive and negatives, called *listwise approach*.[5] As a comparison, the training of reranker is modeled as *pointwise* or *pairwise* optimization, where the model is learned based on a single text or a pair of texts. Due to the different optimization ways, it is difficult to jointly train the two components. To address the above issue, RocketQAv2 [219] proposes a unified listwise learning approach to jointly optimizing a bi-encoder retriever and a cross-encoder reranker. For both retriever and reranker, their relevance predictions are modeled as listwise distributions. To unify the learning process, RocketQAv2 designs a *dynamic listwise distillation* mechanism by distilling the reranker to the retriever. Different from previous distillation methods, it adaptively updates the retriever and reranker by enforcing the interaction of relevance information between the two modules. For optimizing the listwise distillation, RocketQAv2 also employs data augmentation to construct high-quality training data.

*Extension.* In the above subsection, we have discussed how to optimize a retrieval pipeline that integrates a dense retriever and reranker(s). Besides retrieval systems, dense retrievers have been widely used in a variety of application systems that require external knowledge resources, such as open-domain question answering and entity linking. For example, REALM [79] constructs a retrieval-augmented question answering system that can be jointly trained based on asynchronous embedding index update. Other related studies [103, 124, 229] also discuss how to optimize retrieval-augmented systems for open-domain question answering. In general, with dense retrieval techniques, it is easier to develop retrieval-based approaches for downstream tasks. We will discuss the use of dense retrieval techniques for specific applications in Section 8.

## 7 ADVANCED TOPICS

In this section, we discuss several important advanced topics for dense retrieval.

### 7.1 Zero-shot Dense Retrieval

The success of dense retrievers heavily relies on large-scale relevance judgement data. This poses a challenge for a wide range of task scenarios, as it is difficult and expensive to acquire sufficient training corpus when a new domain or task is introduced. In addition, due to large discrepancies across various domains, it becomes challenging for a model trained on a specific dataset to adapt to other domains. Thus, it is important to examine the zero-shot capabilities of dense retrievers, as well as their out-of-distribution performance.

**Zero-shot evaluation datasets.** To our knowledge, BEIR [260] is the first heterogeneous benchmark for examining the zero-shot capabilities of dense retrieval methods. BEIR includes nine

[5]Different from learning to rank [26], here, *listwise* denotes that the optimization is based on an entire candidate list [219].

different retrieval tasks spanning 18 diverse datasets. Based on BEIR, it has been found that a number of bi-encoder dense retrievers that outperform BM25 on in-domain evaluation perform poorly on out-of-distribution evaluation, while cross-attentional reranking models and late interaction models have better zero-shot performance on BEIR. Furthermore, Sciavolino et al. [236] create a dataset containing simple entity-centric questions and find that BM25 significantly outperforms dense retrievers on this dataset. Their analysis shows that dense retrievers perform better on common entities than rare entities, and can also generalize to unseen entities to some extent when question patterns exist in the training data. Liu et al. [147] further measure three kinds of generalization capacities in different settings based on NQ dataset, including training set overlap, compositional generalization, and novel-entity generalization. They find that dense retrievers perform worse in the latter two settings than the first setting.

**Key conclusions of influence factors on zero-shot dense retrieval.** Besides the findings drawn from the overall performance comparison, it is essential to understand how different factors affect the performance of dense retrievers in a zero-shot setting. For this purpose, Ren et al. [220] have conducted a thorough study on the effect of underlying influencing factors on zero-shot retrieval performance based on a total of 12 commonly used retrieval datasets. This study frames the zero-shot retrieval setting by introducing source domain data (available for training) and target domain data (only available for testing). They mainly focus on examining the influence factors from source training corpus (i.e., query set, document set, and data scale), and also discuss other factors such as query type distribution and vocabulary overlap. They empirically find that source training dataset has significant influence on the zero-shot retrieval performance of the dense retrieval models, since only source domain data can be utilized for training. Such an effect can be attributed to a number of specific factors, including vocabulary overlap (*larger is better*), query type distribution (*more comprehensive is better*), and data scale (*more queries are better but not for documents*). Besides, they also find that the dataset bias of the test set potentially affects the performance comparison: Since some datasets are constructed based on lexical matching models [249, 265], they tend to be more favorable for sparse retrieval methods due to a larger term overlap.

**Existing solutions for zero-shot retrieval.** To improve the performance of dense retrievers in the zero-shot setting, there are generally three important approaches explored in the literature [220], including training data augmentation for target domain, term-matching capacity enhancement, and model size scaling. Next, we introduce each approach in detail.

• *Augmenting the target training data.* Considering the lack of labeled data from the target domain, a major solution is to generate large-scale synthetic data for improving the training of dense retrievers. After being trained on large-scale synthetic training data, the zero-shot capabilities of dense retrievers can be improved to some extent. Typically, a data generation model is employed for generating large-scale query-text pairs in the target domain [135, 163, 214, 277]. Recently, Promptagator [52] leverages a large language model consisting of 137 billion parameters, i.e., FLAN [285], to generate large-scale query-text pairs by using only a small number of labeled examples. As an alternative approach, knowledge distillation is commonly adopted for tackling the scarcity of training data, which utilizes a more capable teacher model to improve the student model [37, 259, 277] in zero-shot retrieval scenarios. Besides, several studies [101, 290] conduct unsupervised pretraining by leveraging large-scale positive and negative pairs with different data augmentation methods, e.g., ICT [120] and SimCSE [71].

• *Enhancing the term-matching capability.* Unlike sparse retrievers (e.g., BM25), dense retrievers no longer rely on exact term matching for text retrieval, but instead learn semantic matching for capturing the text relevance. While, empirical studies show that the capability of exact term matching is useful to improve zero-shot retrieval performance [36, 260], since term overlapping is a

strong indicator for text relevance. Thus, sparse retrievers are easier to adapt to zero-shot settings without training data. Inspired by this, several studies propose to enhance the lexical matching capacity of dense retrievers by leveraging sparse retrievers, such as the fusion of rankings [36] or relevance scores [290] for both sparse and dense retrievers. Besides, we can also employ knowledge distillation for improving dense retrievers [37], taking sparse retrievers as the teacher model.

• *Scaling up the model capacity.* Recently, scaling law for language models has been widely explored for raising the performance bar of various tasks. It has been shown useful to improve the zero-shot performance by increasing the model size of dense retrievers. As shown in Reference [187], based on a T5-based dense retriever trained on large-scale question-answer pairs, scaling up the model size with multi-stage training can significantly improve the zero-shot retrieval performance on the BEIR benchmark. Similarly, performance improvement has been observed when the model size is increased from 0.1 billion to 2.4 billion in ERNIE-Search [160].

Besides, Zhan et al. [310] examine how dense retrievers generalize to out-of-domain test queries, called *extrapolation capacity* in their paper. They find that cross-encoder can extrapolate well, but not bi-encoder and sparse retriever. This part can be extended to a more general topic *low-resourced dense retrieval*, and readers can find a comprehensive discussion in a recent survey [241].

### 7.2 Improving the Robustness to Query Variations

Besides the out-of-distribution issue in zero-shot retrieval, dense retrieval models are shown more sensitive to *query variations* than traditional lexical matching methods [38, 196, 333]. Generally, query variations exist widely in real search behaviors, e.g., unexpected query typos due to spelling errors and diverse query formulations for the same information need. We next discuss the effect of query variations on retrieval performance and introduce several enhanced training methods.

**Effect of query variations on retrieval.** There are increasing concerns on the robustness of dense retrieval models to query variations. These studies typically create different types of query variations and examine the performance of dense retrieval models under different query variations. Zhuang et al. [333] present a study on the impact of query typos based on character operations (insertion, deletion, substitution, and swap), showing a significant performance drop for BERT-based retriever and reranker. Penha et al. [196] aim to examine how different types of query variations negatively affect the robustness of the retrieval pipeline. To generate query variations, they consider four syntax-changing types (misspelling, naturality, ordering, and paraphrasing) and two semantics-changing types (gen./specialization, aspect change) for query transformations. Experimental results show that retrieval pipelines are sensitive to query variations (especially the misspelling), which lead to an average 20% drop on performance compared with that evaluated on original queries. Similar findings are reported about the performance drop due to query typos in Reference [38], where they consider eight types of query typos, including new transformations such as adding extra punctuation, stopword removal, and verb-tense shift.

**Enhanced training methods.** To improve the robustness, existing studies propose a series of enhanced training methods that take the query variations into consideration. Basically, these methods utilize data augmentation strategies to incorporate augmented queries in training data, so that dense retrieval models can be aware of query variations during training. Zhuang et al. [333] propose a typos-aware training approach, which changes the queries (with 50% chance) in training set according to different typo types. In this way, the retrieval model is trained with a training set consisting of both original queries and augmented queries with different typo types. Given such an augmentation dataset, Sidiropoulos and Kanoulas [243] propose to construct a query-side contrastive loss: modified queries and randomly sampled queries are considered as positives and negatives, respectively. Based on the work in Reference [333], the same authors [335] further

propose a self-teaching approach by incorporating a distillation loss, and it requires the underlying retrieval model to produce similar rankings under original queries and corresponding variations. They characterize such an idea by reducing the KL-divergence loss between the distributions over the same candidate list for original and augmented queries. Similarly, Chen et al. [38] propose a local ranking alignment method that enforces the original queries and the corresponding variations to produce similar rankings: the probability distributions of the in-batch passages given an original query and its variation should be similar, and the probability distributions over in-batch queries or their variations given a passage should be similar.

Indeed, query variations are not always harmful for retrieval systems. For example, we can utilize *query extension* or *query rewriting* [27] to derive better search results by reformulating the issued query. Furthermore, there are several studies that focus on the model vulnerabilities [145, 283], showing that dense retrieval models are brittle to deliberate attacks. Thus, adversarial training approaches are often used for enhancing the model robustness [145, 283]. We limit our discussion to query variations, more likely to occur in real search scenarios than deliberate attack [196].

### 7.3 Model-based Retrieval

Recently, *model-based retrieval* [180, 222, 258] has been proposed as an alternative paradigm for information retrieval. It essentially learns a parametric model to capture or encode the semantic information of the whole text collection, and further fulfills the retrieval task by predicting the identifiers of relevant documents (i.e., docids) based on the learned model. This paradigm provides a generative viewpoint to understand information retrieval. Since existing model-based retrieval approaches [133, 327] are also developed by pretraining and fine-tuning the Transformer architecture, we include this topic to complement feasible retrieval approaches based on pretrained models.

**The generative scheme.** In essence, model-based retrieval adopts a generative scheme for predicting relevant texts. Indeed, the idea of generative retrieval has been first explored in the task of entity linking [25], where entity identifiers (e.g., unique entity names) are generated conditioned on the text context in an autoregressive manner, and then GRLS [117] extends this idea to long sequence retrieval by introducing multi-step generative retrieval. Specially, the perspective paper [180] envisions the *model-based paradigm* that simplifies the classic *index-retrieve-then-rank* paradigm by a unified relevance model. Further, a representative work [258] called *DSI* develops a generative retrieval model based on a seq2seq encoder-decoder architecture (i.e., T5), where query and docids correspond to input and output, respectively. DSI frames the approach by introducing two key procedures: *indexing* (associating the content of a document with its corresponding docid) and *retrieval* (autoregressively generating docids given the input query). They are essentially related to the two key issues: (i) how to represent the texts with meaningful docids and (ii) how to map a query to relevant docids. Next, we discuss the two issues in detail.

**Representing docids.** Since, in model-based retrieval, we do not directly use the text content for relevance evaluation, we need to design effective docid representations to reflect the underlying semantics of a text. DSI [258] introduces three major docid representation methods, including unstructured atomic identifiers (a unique integer identifier), simple string identifiers (tokenaizable strings) and semantically structured identifiers (clustering-based prefix representation). Intuitively, the second approach is more flexible to use, and in principle one can use various kinds of sequence data to represent a document, e.g., titles and URLs; while the third approach seems to be more meaningful at the cost of additional pre-processing (e.g., clustering based on BERT embeddings), which can be further effectively leveraged by a prefix-based decoding way. DSI shows that the semantically structured identifiers leads to the best performance in its implementation. Later extensions

almost follow the three major methods for representing docids: DynamicRetriever [327] (atomic docid: original docid), SEAL [14] (simple string docid: full ngram signatures with the efficient support of Ferragina Manzini index), NCI [282] (semantic string docid: hierarchical clustering), DSI-QG [332] (simple string docid: generated query), TOME [222] (simple string docid: URL), Ultron [326] (simple string docids: URL and title, semantic string docid: product quantization), GERE [31] (semantic string docids: title and evidence), and CorpusBrain [32] (semantic string docid: title).

**Pretraining for semantic mapping.** The essence of model-based retrieval is to establish the semantic mapping from queries to docids, without an explicit reference to the text content. Thus, in principle, it is more difficult to train generative retrieval models than previous bi-encoder models. For learning query-to-docid mapping, DSI employs two major training strategies, namely, memorization-based pretraining (content mapping) and retrieval-oriented fine-tuning: the former learns to map the content of a document to its corresponding docid and the latter learns to map queries to relevant docids. For content mapping, various pretraining tasks are further proposed to enhance the association between the text content and the docids [32, 258, 326], e.g., mapping a sampled content (e.g., passage, word sets, and ngrams [326, 327]) or associated content (e.g., anchor text [32]) from a text to its docid. Such a strategy can be considered as *learning from pseudo queries*, where various kinds of sampled contents are considered as pseudo queries. Besides, since the amount of available relevance judgement data is usually limited, query generation is useful in these generative retrieval models for enhancing the learning from queries to docids [326, 332], where the original content of a document can be employed to generate potential queries.

**Comparison with traditional models.** Different from traditional retrieval approaches, model-based retrieval simplifies the classic index-retrieve-then-rank paradigm with *a generation-based paradigm*, where a more unified solution to information-retrieval tasks could be potentially developed for various Web resources, such as figures and videos (as envisioned in References [180]). Since such an approach is built on a fully parametric model, it is more easier to be optimized in an end-to-end way. We could potentially get rid of the tedious pipeline training (e.g., jointly optimizing the retriever and reranker) in existing retrieval systems. Further, without using elaborate inverted-index or large-sized embedding index, we do not need to manage a complicated index structure for retrieval, which is easier to use and serve in real-world systems. However, most of model-based retrieval studies only demonstrate the effectiveness of their approaches on MS MARCO subsets or task-specific datasets, while evaluation at a larger scale (e.g., full MS MARCO dataset) is still challenging for model-based retrieval. This emerging retrieval paradigm is still under exploration.

### 7.4 Retrieval-augmented Language Model

In recent years, neural **language models (LM)**, especially PLMs, have proven to be powerful [12, 181, 197, 209] in a variety of tasks. It is shown that PLMs can encode a large amount of semantic knowledge [21, 199] into large-scale model parameters. To enhance the representation capacity, a number of studies explore the scaling law of PLMs [21, 210] by increasing the model size, which makes the training and use of PLMs prohibitively expensive in a resource-limited setting.

To alleviate this issue, retrieval-augmented LMs have been proposed by allowing the model to explicitly access external data [19, 79, 83, 112, 300]. Retrieval-augmented models tend to be more parameter efficient, since they retrieve the knowledge rather than fully storing it in model parameters. Khandelwal et al. [112] present $k$NN-LM, an approach that linearly interpolates between the LM's next word distribution and the next word distribution of $k$-nearest-neighbor model. The experiments demonstrate that $k$NN-LM works particularly well when predicting rare patterns, since

it allows explicit retrieval of rare patterns. Furthermore, Yogatama et al. [300] incorporate a gating mechanism to replace the fixed interpolation parameter in $k$NN-LM, which adaptively combines short-term memory with long-term memory (i.e., knowledge retrieved from external memory) for prediction conditioned on the context.

Since external knowledge resource is very important to retrieval-augmented LMs, Borgeaud et al. [19] further explore scaling up the retrieval corpus to trillions of tokens for large LMs ranging from 150M to 7B parameters, which leads to an improved capacity of LMs. In these studies, LMs are frozen pretrained retrievers. As a comparison, Guu et al. [79] propose REALM to jointly optimize the knowledge retriever and the knowledge-augmented encoder (detailed in Section 4.5.2).

## 8 APPLICATIONS

In this section, we review the applications of dense retrieval technique in three aspects, namely, the applications to different retrieval settings and industry practice.

### 8.1 Specific Retrieval Scenarios

In this part, we describe several specific retrieval settings and corresponding retrieval approaches.

*8.1.1 Temporal Retrieval.* Most of existing works conduct the study of dense retrieval on synchronic document collections (e.g., Wikipedia). However, there are a large proportion of temporal-dependent questions in real application scenarios. To advance the research on temporal retrieval, Zhang et al. [315] create an open-retrieval question answering dataset called *SituatedQA*, where the systems are required to answer questions given the temporal or geographical contexts. Their experimental results show that existing dense retrievers cannot produce satisfying answers that are frequently updated. Besides, Wang et al. [273] create another large question answering dataset called *ArchivalQA*, consisting of 1,067,056 question-answer pairs, for temporal news question answering. These datasets can be used to develop time-sensitive dense retrievers.

*8.1.2 Structured Data Retrieval.* Previous sections mainly review the progress of dense retrieval on unstructured text. Recently, dense retrieval has also been applied to structured data, e.g., tables and knowledge bases. Herzig et al. [86] apply a bi-encoder-based dense retriever on table retrieval, which is a Transformer-based language model pretrained on millions of tables with the awareness of tabular structure. As shown in Reference [86], the proposed approach outperforms BM25 by more than 40 percentage points on the NQ-TABLES dataset. Oguz et al. [192] further study the problem of unifying open-domain question answering with both structured and unstructured data. They first linearize tables and knowledge bases into texts, and then train a bi-encoder-based dense retriever on multiple datasets consisting of texts, lists, tables, and knowledge bases. The experimental results show that the proposed approach performs well on both entity-oriented and text-oriented benchmarks. Kostic et al. [115] also explore the unified retrieval by proposing a tri-encoder, with one unique encoder each for question, text and table, respectively, showing that the tri-encoder performs better than the bi-encoder with one encoder for the question, the other one for both text and tables.

*8.1.3 Multilingual Retrieval.* In the literature, monolingual retrieval (mainly in English) is most commonly studied, since available large-scale labeled datasets are mainly in English (e.g., MS MARCO and NQ). To facilitate the research on other languages rather than English, Bonifacio et al. [18] create a multilingual version of the MS MARCO passage ranking dataset using machine translation, called *mMARCO*, consisting of 13 languages. The experimental results show that the models fine-tuned on multilingual datasets perform better than that fine-tuned only on the original English dataset. Zhang et al. [317] present the multi-lingual benchmark dataset called *Mr. TYDI*,

consisting of monolingual retrieval corpus in 11 languages. They show that BM25 performs better than a multi-lingual adaptation of DPR on this benchmark dataset. The above multilingual retrieval settings aim to find the answer from the corpus in the same language as the question. However, many languages face the scarcity of text resources, where there are few reference texts written in a specific language. Hence, Asai et al. [6] propose a cross-lingual task, called **Cross-lingual Open Retrieval Question Answering (XOR QA)**, that requires answering questions in one language based on a text in another language. They create a large-scale dataset consisting of 40K questions across seven non-English languages for this proposed task. The proposed approach relies on machine translation modules for question translation and answer translation. Since the multi-step machine translations (e.g., query translation and answer translation) in previous approaches may lead to error propagation, Asai et al. [7] further propose a multilingual dense passage retriever that extends DPR to retrieve candidate passages from multilingual document collections in a cross-lingual way. It adopts an iterative training approach to fine-tuning a multilingual PLM (e.g., mBERT [201]), and achieves performance improvement in a number of languages.

*8.1.4 Other Retrieval Tasks.* So far, dense retrieval has been widely applied to various domain-specific retrieval tasks, such as mathematical IR [320], code retrieval [158], and biomedical IR [162]. In essence, when considering a specific retrieval task, we need to adapt the dense retrievers to effectively fit domain-specific text data (continual pretraining is often needed), deal with special content features or formats (e.g., formulas or code) and resolve other possible training issues (e.g., lack of training data). Besides, another important topic is cross-modal retrieval [278], where we consider a retrieval setting across different modalities (e.g., text-to-image retrieval and text-to-video retrieval). The key to these cross-modal tasks is to establish the fusion or mapping between different modalities. With the development of vision-language pretraining models exemplified by CLIP [208], the capacity of learning shared representations across modalities has been largely enhanced, which can improve the downstream cross-modal retrieval tasks. While, this survey is mainly focused on text resources for retrieval, and interested readers can refer to Reference [278] for a detailed discussion.

### 8.2 Industrial Practice

Besides research-purpose studies, considerable efforts have attempted to deploy dense retrieval techniques in real systems. We review these works in three typical applications as follows.

In Web search, the term mismatch between search queries and web pages is one of the major challenges in practice. For enhancing the semantic matching capacity, Liu et al. [151] propose a multi-stage training approach to optimizing dense retriever by leveraging both large-scale weakly supervised training data (e.g., search logs) and small-scale human labeled training data. Such a multi-stage training approach leads to significant improvement compared to single-stage training. Besides, since dense retrieval is less capable in lexical matching, an integration of dense and sparse retrieval techniques has been employed in Baidu search [151] and Spotify search [255].

In sponsor search [205], the systems should optimize multiple objectives simultaneously, e.g., **query-advertisement (ad)** relevance and **clicked-through rate (CTR)**. Under a traditional retrieval pipeline, the retriever first retrieves the relevant ads by only considering the relevance between queries and ads, and then the reranker reorders the retrieved ads according to the CTR criteron. Fan et al. [56] argue that the separation of relevance learning and CTR prediction tends to produce sub-optimal performance, and they propose to train a retriever that jointly considers both relevance and CTR. Specifically, they train the retriever from the clicked logs augmented by a relevance teacher. The proposed system is called *Mobius* and has been deployed into Baidu's sponsor search. As another solution, Zhang et al. [313] propose to unify knowledge distillation

and contrastive learning for training a more capable retriever, namely, *Uni-retriever*. Specifically, the retriever learns to retrieve high-relevance ads by distilling the knowledge from a relevance teacher, while learns to retrieve high-CTR ads by contrastive learning. It has been reported that Uni-Retriever has been included as one of the major multi-path retrievers in Bing's sponsor search [313]. Besides, Dodla et al. [55] fuse dense retrievers and non-autoregressive generators in a multi-task setting (sharing the same encoder), showing a significant performance gain in bid keyword retrieval.

In personalized search and product search, the personalized features and product features should be incorporated into the retrievers. Huang et al. [95] incorporate the embeddings of search context (social relation, location, etc.) as the input of dense retriever. Liu et al. [153] incorporate the embeddings of product images as the input of dense retriever. In these systems, Huang et al. [95] and Li et al. [130] also find that hard negatives are important for training dense retrievers (as we discussed in Section 4.2). Magnani et al. [171] design a hybrid dense-sparse system for e-commerce search deployed at Walmart, and they select negatives based on three strategies including product type matching, query token matching and MonoBERT-based scoring.

## 9 CONCLUSION AND REMAINING ISSUES

In this survey, we thoroughly review the recent progress of dense retrieval based on PLMs. As an important evolution of language intelligence techniques, PLMs empower dense retrieval models with excellent modeling capacities to capture and represent text semantics for relevance matching. Our survey has extensively discussed the key issues and the mainstream solutions in four major aspects to develop dense retrieval systems, including architecture, training, indexing and integration. Next, we briefly summarize the discussions of this survey and introduce some remaining issues for dense retrieval.

**PLM-based architecture.** We have reviewed two kinds of PLM architectures for dense retrieval, namely, bi-encoder (*efficient* yet *less effective*) and cross-encoder (*effective* yet *less efficient*), which are often used to implement the retriever and reranker, respectively. Since the bi-encoder has limited capacity in capturing the semantic interaction between query and text, the multi-representation technique (i.e., learning multiple contextual representations of a text) has been widely explored in enhancing the fine-grained interaction modeling. Although dense retrieval models have achieved decent retrieval performance on several benchmark datasets, it has been shown that their capacities in some specific settings are rather limited, especially the capability on zero-shot retrieval [260]. Compared to sparse retrievers (e.g., BM25), dense retrievers perform better for semantic matching, but less well for lexical matching [111, 236]. Thus, there has been increasing attention in combining the merits of both dense retrievers and sparse retrievers in a unified retrieval system [37, 139]. Moreover, dense retrieval models rely on latent semantic representations for relevance matching. It is useful to explore more interpretable mechanisms to understand how dense retrieval models behave and perform in response to different types of queries, e.g., how they mimic "*bag-of-words*" retrieval [161]. In addition, instead of adapting a dense retriever to some specific dataset, it is meaningful to develop more general retrievers over multiple domains, and specific architecture designs (e.g., mixture of experts [61]) would be worth further exploration for retrieval models. Furthermore, in some cases, dense retrievers need to process very long text, and length-extrapolatable techniques [252] are required to facilitate the accurate modeling of the text semantics.

**Training approach.** Compared with previous neural IR methods, PLM-based dense retrieval models are more difficult to be optimized given the large number of parameters. Even with the

ever-increasing labeled dataset, it still requires specific training techniques to effectively optimize large retrieval models. We have summarized three major issues to address for improving dense retrieval systems, i.e., large-scale candidate space, limited relevance judgements, and pretraining discrepancy. Focused on the three issues, the researchers have made extensive efforts by designing various effective training strategies, largely raising the performance bar of dense retrievers on benchmark datasets [65, 111, 207, 289]. In particular, negative selection is a key step in the training approach, and using *more* negatives (e.g., in- and cross-batch negatives) or *more high-quality* negatives (e.g., sampling hard negatives) typically lead to a better retrieval performance. Besides, to enhance the capacity of bi-encoder, it is very useful to distill a more capable teacher model (e.g., cross-encoder) and design suitable pretraining tasks (e.g., representation-enhanced pretraining). More recently, auto-encoder and contrastive learning-based pretraining methods have been widely adopted to enhance the semantic representations of texts. As another improvement direction, it is important to enhance the performance of dense retrievers under low-resourced setting (e.g., zero-shot retrieval), and data- and parameter-efficient training approaches are useful for enhancing the generalization capacity of dense retrievers. Furthermore, transfer learning can be also employed to mitigate the scarcity of labeled data by leveraging data-rich domains.

**Dense vector index.** Without the support of a suitable index structure, the search process (i.e., finding the most close text vectors w.r.t. query vectors) would be extremely slow for dense retrieval. To tackle this issue, ANNS algorithms [74, 108, 242] have been widely used, achieving highly efficient retrieval performance (e.g., sublinear time cost [242]). Compared with term-based index, it is more difficult to maintain the ANNS index for frequent operations of adding, deleting and updating the text embeddings. For example, when the optimization involves in the update of indexed embeddings, it usually suffers from the *index staleness* issue [79]. Therefore, it is necessary to develop efficient online update algorithms for enhancing the flexibility of ANNS index. Besides the effectiveness, the cost for maintenance and use is also important for the practical deployment of dense retrieval systems [90]. In practice, quantization techniques have become a major technical approach to alleviate the cost of dense vector index. To fit quantization for the retrieval task, it is important develop effective approaches to optimize quantization-based vector index, which become very difficult due to the involved non-differential operation [306, 312]. Overall, the dense vector index should be well suited to both the architecture and optimization of the PLM-based retrievers, and more research is still required in this line.

**Retrieval pipeline.** An information-retrieval system typically adopts a pipeline way (consisting of first-stage retrieval and reranking stages) to recall and rank the texts, which gradually reduces the search space and refines the top-ranked texts. Generally, it is more complicated to optimize the entire retrieval pipeline than a single-stage component in it (e.g., first-stage retriever), though a joint optimization approach is preferred. As the major difficulty for joint training, these stages are usually learned according to different optimization goals, based on varied input and output. In early studies [191], the multiple stages in a retrieval pipeline are separately optimized. Recent studies propose a series of improved optimization approaches for jointly training the retrieval pipeline [79, 219]. The basic idea is to let the retriever and the reranker adjust according to the learned relevance information from each other. However, it is still challenging to optimize a multi-stage retrieval pipeline, especially when it is built in a sparse-dense hybrid way. Besides, it is also meaningful to study how to automatically construct the pipeline (e.g., what components should be used for different stages) by learning from labeled dataset. Furthermore, to reduce the maintenance cost, it is also useful to develop effective approaches (e.g., distillation) to simplify the multi-stage ranking pipeline by merging consecutive ranking components.

Recently, **large language models (LLMs)** [21, 319] have shown more superior performance than previous relatively small PLMs, which has inspired more thoughts on the research paradigm and technical approach of information retrieval. Despite the powerful capacities, LLMs are mainly focused on generation-based task solving, and also incur a higher cost on tuning and inference, making it difficult to fully substitute for PLMs for developing dense retrievers. Overall, PLM-based dense retrievers and LLMs bear reciprocal or mutual relations in recent research [329]. On the one hand, LLMs are powerful in possessing rich world knowledge and various abilities, which are beneficial to enhance dense retrievers in multiple aspects, including complex semantic understanding [177], ranking refinement [251], search result extraction [291], and low-sourced domain adaptation [17]. On the other hand, LLMs still rely on external information resources when dealing with domain-specific tasks [299] or those outside its knowledge scope [195], where dense retrievers have become necessary tools for finding the relevant evidence [221] or the demonstration examples [212]. Generally, both research directions (i.e., LLMs for search and search for LLMs) are promising to explore for improving the quality of information-seeking systems for human users.

## APPENDICES

## A SUPPLEMENTARY CONTENT

In this section, we introduce the available datasets, the evaluation metrics, and the supporting libraries for dense retrieval.

Table 4. Detailed Statistics of Available Retrieval Datasets

| Categorization | Domain | Dataset | #q in train | #label in train | #q in dev | #q in test | #instance |
|---|---|---|---|---|---|---|---|
| Information retrieval | Web | MS MARCO [185] | 502,939 | 532,761 | 6,980 | 6,837 | 8,841,823 |
| | Web | mMARCO [18] | 808,731 | — | 101,093 | — | 8,841,823 |
| | News | TREC-NEWS [246] | — | — | — | 57 | 594,977 |
| | Biomedical | TREC-COVID [224] | — | — | — | 50 | 171,332 |
| | Biomedical | NFCorpus [20] | 5,922 | 110,575 | 324 | 323 | 3,633 |
| | Twitter | Signal-1M [249] | — | — | — | 97 | 2,866,316 |
| | Argument | Touché-2020 [16] | — | — | — | 249 | 528,155 |
| | Argument | ArguAna [271] | — | — | — | 1,406 | 8,674 |
| | Wikipedia | DBPedia [82] | — | — | 67 | 400 | 4,635,922 |
| | Web | ORCAS [43] | 10.4M | 18.8M | — | — | 3,213,835 |
| | Wikipedia | EntityQuestions [236] | 176,560 | 186,367 | 22,068 | 22,075 | — |
| | Web | MS MARCO v2 [45] | 277,144 | 284,212 | 8,184 | — | 138,364,198 |
| | Web | $\text{DuReader}_{\text{retrieval}}$ [206] | 97,343 | 86,395 | 2,000 | 8,948 | 8,096,668 |
| Question answering | Wikipedia | Natural Questions [116] | 152,148 | 152,148 | 6,515 | 3,610 | 2,681,468 |
| | Wikipedia | SQuAD [211] | 78,713 | 78,713 | 8,886 | 10,570 | 23,215 |
| | Wikipedia | TriviaQA [109] | 78,785 | 78,785 | 8,837 | 11,313 | 740K |
| | Wikipedia | HotpotQA [298] | 85,000 | 170,000 | 5,447 | 7,405 | 5,233,329 |
| | Web | WebQuestions [13] | 3,417 | 3,417 | 361 | 2,032 | — |
| | Web | CuratedTREC [11] | 1,353 | 1,353 | 133 | 694 | — |
| | Finance | FiQA-2018 [172] | 5,500 | 14,166 | 500 | 648 | 57,638 |
| | Biomedical | BioASQ [265] | 3,743 | 35,285 | — | 497 | 15,559,157 |
| | StackEx. | CQADupStack [94] | — | — | — | 13,145 | 457,199 |
| | Quora | Quora [100] | — | — | 5,000 | 10,000 | 522,931 |
| | News | ArchivalQA [273] | 853,644 | 853,644 | 106,706 | 106,706 | 483,604 |
| | Web | CCQA [97] | 55M | 130M | — | — | — |
| Other tasks | Wikipedia | FEVER [261] | — | 140,085 | 6,666 | 6,666 | 5,416,568 |
| | Wikipedia | Climate-FEVER [54] | — | — | — | 1,535 | 5,416,593 |
| | Scitific | SciFact [272] | 809 | 920 | — | 300 | 5,183 |
| | Scitific | SciDocs [41] | — | — | — | 1,000 | 25,657 |

Here, "q" is the abbreviation of queries, and "instance" denotes a candidate text in the collection.

### A.1 Datasets

Compared with traditional retrieval models, dense retrieval models are more data hungry, requiring large-scale labeled datasets to learn the parameters of the PLMs. In recent years, there are a number of retrieval datasets with relevance judgements released publicly, which significantly advances the research of dense retrieval. We categorize the available retrieval datasets into three major categories according to their original tasks, namely, information retrieval, question answering, and other task settings. The statistics of the available datasets are shown in Table 4.

As we can see from Table 4, Wikipedia and Web are two major resources for creating these datasets. Among these datasets, *MS MARCO* dataset [185] contains a large amount of queries with annotated relevant passages in Web documents, and NQ [116] dataset contains Google search queries and documents with paragraphs and answer spans from the top-ranked Wikipedia pages. Among these datasets, MS MARCO and NQ datasets have been widely used for evaluating dense retrieval models. A recent study [48] has summarized the promotion effect of MS MARCO on the progress of dense retrieval: Neural models can explore large-scale data for training. Besides, based on MS MARCO, several variants have been created to enrich the evaluation characteristics on some specific aspect, e.g., the multilingual version mMARCO [18] and the MS MARCO Chameleons dataset [3] (consisting of obstinate queries that are difficult to answer by neural retrievers and have similar query length and distribution of relevance judgements with easier queries). Besides, Sciavolino et al. [236] create EntityQuestions dataset, as a challenging test set for the models trained on NQ, which contains simple factoid questions about entities from Wikipedia. Another interesting observation is that there are increasingly more domain-specific retrieval datasets, including COVID-19 pandemic dataset [224], financial dataset [172], biomedical dataset [265], climate-specific dataset [54], and scientific dataset [272].

Besides the presented datasets in Table 4, several more comprehensive benchmark datasets are released to evaluate the overall retrieval capability of the retrieval models by aggregating representative datasets and conducting diverse evaluation tasks, such as *BEIR* [260] and *KILT* [198].

Although existing datasets largely improve the training of dense retrievers, in these datasets, a query typically corresponds to very few relevance judgements. For example, Nogueira et al. observe that most of queries in MS MARCO dataset contains only one labeled positive[4], which is likely to be smaller than the actual number of relevant ones in the collection. It is mainly because it is time-consuming to construct complete relevance judgements for a large dataset. The incomplete relevance annotations will lead to several training issues such as false negative, which potentially affects the retrieval performance.

Besides, to date, most of the released datasets are created in English, and it is more difficult to obtain sufficient labeled data for training a non-English dense retriever. More recently, DuReader-retrieval [206] releases a large-scale Chinese dataset consisting of 90K queries from Baidu search and over 8M passages for passage retrieval. To enhance the evaluation quality, DuReader-retrieval tries to reduce the false negatives in development and testing sets, and also removes the training queries that are semantically similar to the development and testing queries. Besides, DuReader-retrieval provides human-translated queries (in English) for cross-lingual retrieval.

### A.2 Evaluation Metrics

To evaluate the retrieval capacity of an information-retrieval system, a number of factors need to be considered [81, 81, 267]: effectiveness, efficiency, diversity, novelty, and so on. This survey mainly focuses on the effectiveness for the retrieval system.[6] We next introduce the commonly

[6]Note that these metrics can be used in both first-stage retrieval and reranking. However, we only describe the evaluation metrics from a general ranking perspective without considering specific task scenarios.

used evaluation metrics for ranking, including Recall, Precision, MAP, MRR, and NDCG. For dense retrieval tasks, top-ranked texts are more important for evaluation, and therefore cut-off metrics are often adopted to examine the quality of texts at top positions. Next, we introduce several commonly used metrics for dense retrieval in detail.

In the traditional retrieval benchmarks, Recall is the fraction of relevant texts that are actually retrieved by a retrieval model among all the relevant ones, and Recall@$k$ [328] calculates a truncated Recall value at the $k$th position of a retrieved list:

$$\text{Recall@}k = \frac{1}{|Q|}\sum_{q=1}^{|Q|}\frac{\#\text{retr}_{q,k}}{\#\text{rel}_q}, \tag{11}$$

where $\#\text{retr}_{q,k}$ denotes the number of relevant texts w.r.t. query $q$ retrieved at top $k$ positions by a retrieval method, and $\#\text{rel}_q$ denotes the total number of relevant texts for query $q$. Here, we average the Recall@$k$ values over the queries from the query set $Q$.

In dense retrieval, there is a commonly used metric, *Top-k Accuracy*, and it computes the proportion of queries for which the top-$k$ retrieved texts contain the answers [111], defined as

$$\text{Accuracy@}k = \frac{1}{|Q|}\sum_{q=1}^{|Q|}\mathbb{I}(\#\text{retr}_{q,k} > 0), \tag{12}$$

where $\mathbb{I}(\cdot)$ is an binary indicator function that only returns 1 when the case is true. In contrast to traditional TREC benchmarks, mainstream dense retrieval benchmarks, such as NQ, aim to find answers to queries instead of retrieving all relevant texts. According to References [111, 207], a retrieved list is considered to *accurately* solve a query when it contains the answer, not necessarily retrieving all the relevant texts.[7]

Besides, Precision@$k$ [328] calculates the average proportion of relevant texts among the top $k$ positions over the query set $Q$:

$$\text{Precision@}k = \frac{1}{|Q|}\sum_{q=1}^{|Q|}\frac{\#\text{retr}_{q,k}}{k}. \tag{13}$$

Based on Precision@$k$, the **Average Precision (AP)** further averages the precision values at the positions of each positive text for a query:

$$\text{AP}_q = \frac{1}{\#\text{rel}_q}\sum_{k=1}^{L}\text{Precision@}l \times \mathbb{I}(q,k), \tag{14}$$

where Precision@$l$ is the per-query version of the precision value at the $l$th position, $L$ is the length of a retrieved list and $\mathbb{I}(q,l)$ is an indicator function returning 1 only when the $l$th position corresponds to a relevant text for query $q$. Furthermore, MAP [328] calculates the average Precision scores over a set of queries $Q$:

$$\text{MAP} = \frac{1}{|Q|}\sum_{q=1}^{|Q|}\text{AP}_q. \tag{15}$$

Not only counting the occurrence of positive texts, Normalized **Discounted Cumulative Gain (DCG)** [104] further incorporates the position of a relevant text into consideration: it prefers a

[7]Note that Accuracy@$k$ is also known as Recall@$k$ in some dense retrieval studies [66, 207], which is somehow different from the definition used in traditional retrieval benchmarks.

ranking that places a relevant text at a higher position:

$$\text{DCG}_q@k = \sum_{i=1}^{k} \frac{2^{g_i} - 1}{\log_2(i+1)}, \tag{16}$$

where $g_i$ is the graded relevance score for the $i$th retrieved text. Based on the above definition of DCG, NDCG is the sum of the normalized DCG values at a particular rank position:

$$\text{nDCG}@k = \frac{1}{|Q|} \sum_{q=1}^{|Q|} \frac{\text{DCG}_q@k}{\text{IDCG}_q@k}, \tag{17}$$

where DCG@$k$ and IDCG@$k$ denote discounted cumulative gain and ideal discounted cumulative gain at a particular rank position $k$, respectively.

Furthermore, MRR [270] averages the reciprocal of the rank of the first retrieved positive text over a set of queries $Q$:

$$\text{MRR} = \frac{1}{|Q|} \sum_{q=1}^{|Q|} \frac{1}{\text{rank}_q}, \tag{18}$$

where $\text{rank}_q$ is the position of the first retrieved positive text w.r.t. query $q$.

For first-stage retrieval, Recall@$k$ and Accuracy@$k$ are the most commonly used metrics, since its main focus is to recall as many relevant texts or answers as possible at a truncation length; while for ranking, MRR, NDCG, and MAP are more commonly used in practice. For a comprehensive discussion about IR evaluation, the readers are suggested to read more focused references [81, 176, 234, 267].

### A.3 Code Library Resource

Recently, several open-sourced dense retrieval libraries have been released for research purpose. As a representative library, Tevatron [69] has developed a modularized framework for building dense retrieval models based on PLMs via command-line interfaces. It supports a number of important procedures involved in a complete retrieval pipeline including text processing, model training, text encoding, and text retrieval. It can be accessed at the link: https://github.com/TextTron/Tevatron.

Besides, Pyserini [136] is a toolkit that is designed to facilitate reproducible research for information retrieval. Specifically, it supports both sparse retrieval and dense retrieval with Anserini IR toolkit [296] and FAISS [108]. It also provides the evaluation scripts for the standard IR test collections. It can be accessed from the link: http://pyserini.io/.

SentenceTransformers [215] is another library that provides an easy way to compute dense embeddings for sentences and paragraphs based on Transformer-based networks. Specifically, it integrates the implementation of Sentence-BERT [215] and TSDAE [276]. It can be accessed at the link: https://www.sbert.net/.

To enhance the validation of dense retriever checkpoints, Asyncval [334] is released to ease and accelerate the checkpoint validation for dense retrieval models. An important merit is that the training can be decoupled from checkpoint validation with Asyncval. It can be accessible at the link: https://github.com/ielab/asyncval.

OpenMatch [155] has been originally proposed for neural information retrieval (v1), and extended (v2) to support dense retrieval on commonly used benchmarks such as MS MARCO and NQ. It can be accessed at the link: https://github.com/thunlp/OpenMatch.

MatchZoo [76] is a text matching library that supports a number of neural text matching models, and allow users to develop new models by providing rich interfaces. It can be accessed at the link: https://github.com/NTMC-Community/MatchZoo.

As the supporting resource, we also release an open-sourced implementation of dense retrievers based on our previous work RocketQA [207], at the link: https://github.com/PaddlePaddle/RocketQA, including RocketQA [207], RocketQA$_{PAIR}$ [218], and RocketQAv2 [219]. This software also provides an easy-to-use toolkit with pre-built models (including both English and Chinese models) for direct use after the installation. We also aggregate other open-sourced codes for related dense retrieval papers in our survey site, which can be found in the link: https://github.com/RUCAIBox/DenseRetrieval.

## ACKNOWLEDGMENTS

The authors gratefully appreciate the anonymous reviewers for their valuable and detailed comments, which greatly helped to improve the quality of this article.

## REFERENCES

[1] Akiko Aizawa. 2003. An information-theoretic perspective of TF–IDF measures. *Info. Process. Manage.* 39, 1 (2003), 45–65.

[2] Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. 2019. Synthetic QA corpora generation with roundtrip consistency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 6168–6173.

[3] Negar Arabzadeh, Bhaskar Mitra, and Ebrahim Bagheri. 2021. MS MARCO chameleons: Challenging the MS MARCO leaderboard with extremely obstinate queries. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*. 4426–4435.

[4] Negar Arabzadeh, Alexandra Vtyurina, Xinyi Yan, and Charles L. A. Clarke. 2021. Shallow pooling for sparse labels. Retrieved from https://arXiv:2109.00062

[5] Negar Arabzadeh, Xinyi Yan, and Charles L. A. Clarke. 2021. Predicting efficiency/effectiveness trade-offs for dense vs. sparse retrieval strategy selection. Retrieved from https://arXiv:2109.10739

[6] Akari Asai, Jungo Kasai, Jonathan Clark, Kenton Lee, Eunsol Choi, and Hannaneh Hajishirzi. 2021. XOR QA: Cross-lingual open-retrieval question answering. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 547–564.

[7] Akari Asai, Xinyan Yu, Jungo Kasai, and Hanna Hajishirzi. 2021. One question answering model for many languages with cross-lingual dense passage retrieval. *Adv. Neural Info. Process. Syst.* 34 (2021).

[8] Martin Aumüller, Erik Bernhardsson, and Alexander Faithfull. 2017. ANN-benchmarks: A benchmarking tool for approximate nearest-neighbor algorithms. In *Proceedings of the International Conference on Similarity Search and Applications (SISAP'17)*.

[9] Ricardo Baeza-Yates and Berthier A. Ribeiro-Neto. 2011. *Modern Information Retrieval—The Concepts and Technology Behind Search, 2nd Ed.*

[10] Vidhisha Balachandran, Ashish Vaswani, Yulia Tsvetkov, and Niki Parmar. 2021. Simple and efficient ways to improve REALM. Retrieved from https://arXiv:2104.08710

[11] Petr Baudiš and Jan Šedivỳ. 2015. Modeling of the question answering task in the YodaQA system. In *Proceedings of the International Conference of the Cross-language Evaluation Forum for European Languages*. Springer, 222–228.

[12] Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2000. A neural probabilistic language model. In *Proceedings of the Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS'00)*. 932–938.

[13] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 1533–1544.

[14] Michele Bevilacqua, Giuseppe Ottaviano, Patrick Lewis, Wen tau Yih, Sebastian Riedel, and Fabio Petroni. 2022. Autoregressive search engines: Generating substrings as document identifiers. Retrieved from https://abs/2204.10628

[15] Nitin Bhatia and Vandana. 2010. Survey of nearest-neighbor techniques. Retrieved from https://arXiv:1007.0085

[16] Alexander Bondarenko, Maik Fröbe, Meriem Beloucif, Lukas Gienapp, Yamen Ajjour, Alexander Panchenko, Chris Biemann, Benno Stein, Henning Wachsmuth, Martin Potthast, and Matthias Hagen. 2020. Overview of touché 2020: Argument retrieval. In *Proceedings of the Conference and Labs of the Evaluation Forum (CLEF'20)*.

[17] Luiz Bonifacio, Hugo Abonizio, Marzieh Fadaee, and Rodrigo Nogueira. 2022. Inpars: Data augmentation for information retrieval using large language models. Retrieved from https://arXiv:2202.05144
[18] Luiz Henrique Bonifacio, Israel Campiotti, Roberto Lotufo, and Rodrigo Nogueira. 2021. mMARCO: A multilingual version of MS MARCO passage ranking dataset. Retrieved from https://arXiv:2108.13897
[19] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, T. W. Hennigan, Saffron Huang, Lorenzo Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and L. Sifre. 2021. Improving language models by retrieving from trillions of tokens. Retrieved from https://abs/2112.04426
[20] Vera Boteva, Demian Gholipour, Artem Sokolov, and Stefan Riezler. 2016. A full-text learning to rank dataset for medical information retrieval. In *Proceedings of the European Conference on Information Retrieval*. Springer, 716–722.
[21] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford,Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Proceedings of the Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems (NeurIPS'20)*.
[22] Yinqiong Cai, Yixing Fan, Jiafeng Guo, Fei Sun, Ruqing Zhang, and Xueqi Cheng. 2021. Semantic models for the first-stage retrieval: A comprehensive review. Retrieved from https://arXiv:2103.04831
[23] Yinqiong Cai, Jiafeng Guo, Yixing Fan, Qingyao Ai, Ruqing Zhang, and Xueqi Cheng. 2022. Hard negatives or false negatives: Correcting pooling bias in training neural ranking models. Retrieved from https://arXiv:2209.05072
[24] Arthur Câmara and Claudia Hauff. 2020. Diagnosing BERT with retrieval heuristics. In *Proceedings of the Advances in Information Retrieval 42nd European Conference on IR Research (ECIR'20)*, Vol. 12035. 605–618.
[25] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive entity retrieval. Retrieved from https://abs/2010.00904
[26] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: From pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning (ICML'07)*. 129–136.
[27] Claudio Carpineto and Giovanni Romano. 2012. A survey of automatic query expansion in information retrieval. *ACM Comput. Surveys* 44, 1 (2012), 1–50.
[28] Wei-Cheng Chang, Felix X. Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. 2020. Pre-training tasks for embedding-based large-scale retrieval. In *Proceedings of the 8th International Conference on Learning Representations (ICLR'20)*.
[29] 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*. 1870–1879.
[30] Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. 2017. A survey on dialogue systems: Recent advances and new frontiers. *SIGKDD Explor.* 19, 2 (2017), 25–35.
[31] Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Yixing Fan, and Xueqi Cheng. 2022. GERE: Generative evidence retrieval for fact verification. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'22)*. 2184–2189.
[32] Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Yiqun Liu, Yixing Fan, and Xueqi Cheng. 2022. CorpusBrain: Pre-train a generative retrieval model for knowledge-intensive language tasks. Retrieved from https://abs/2208.07652
[33] Qi Chen, Haidong Wang, Mingqin Li, Gang Ren, Scarlett Li, Jeffery Zhu, Jason Li, Chuanjie Liu, Lintao Zhang, and Jingdong Wang. 2018. *SPTAG: A Library for Fast Approximate Nearest Neighbor Search.*
[34] Qi Chen, Bing Zhao, Haidong Wang, Mingqin Li, Chuanjie Liu, Zengzhong Li, Mao Yang, and Jingdong Wang. 2021. SPANN: Highly efficient billion-scale approximate nearest-neighbor search. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS'21)*.
[35] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning (ICML'20)*. 1597–1607.
[36] Tao Chen, Mingyang Zhang, Jing Lu, Michael Bendersky, and Marc-Alexander Najork. 2022. Out-of-domain semantics to the rescue! zero-shot hybrid retrieval models. In *Proceedings of the Advances in Information Retrieval 44th European Conference on IR Research (ECIR'22)*.
[37] Xilun Chen, Kushal Lakhotia, Barlas Oğuz, Anchit Gupta, Patrick Lewis, Stan Peshterliev, Yashar Mehdad, Sonal Gupta, and Wen-tau Yih. 2021. Salient phrase aware dense retrieval: Can a dense retriever imitate a sparse one? Retrieved from https://arXiv:2110.06918

[38] Xuanang Chen, Jian Luo, Ben He, Le Sun, and Yingfei Sun. 2022. Towards robust dense retrieval via local ranking alignment. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI'22)*. 1980–1986.
[39] Hao Cheng, Hao Fang, Xiaodong Liu, and Jianfeng Gao. 2022. Task-aware specialization for efficient and robust dense retrieval for open-domain question answering. Retrieved from https://arXiv:2210.05156
[40] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: encoder–decoder approaches. In *Proceedings of the 8th Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST'14)*. 103–111.
[41] Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel Weld. 2020. SPECTER: Document-level representation learning using citation-informed transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2270–2282.
[42] Alexis Conneau and Douwe Kiela. 2018. SentEval: An evaluation toolkit for universal sentence representations. In *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC'18)*.
[43] Nick Craswell, Daniel Campos, Bhaskar Mitra, Emine Yilmaz, and Bodo Billerbeck. 2020. ORCAS: 20 million clicked query-document pairs for analyzing search. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM'20)*. 2983–2989.
[44] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the TREC 2020 deep learning track. Retrieved from https://arXiv:2102.07662
[45] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Jimmy Lin. [n.d.]. TREC 2021 Deep Learning Track Guidelines. Retrieved from https://microsoft.github.io/msmarco/TREC-Deep-Learning.html
[46] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Jimmy Lin. 2022. Overview of the TREC 2021 deep learning track. In *Proceedings of the Text Retrieval Conference (TREC'22)*.
[47] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. Overview of the TREC 2019 deep learning track. Retrieved from https://arXiv:2003.07820
[48] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Fernando Campos, and Jimmy J. Lin. 2021. MS MARCO: Benchmarking ranking models in the large-data regime. *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*.
[49] Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. In *Proceedings of the Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems (NeurIPS'15)*. 3079–3087.
[50] Zhuyun Dai and Jamie Callan. 2020. Context-aware document term weighting for ad hoc search. In *Proceedings of the Web Conference (WWW'20)*. 1897–1907.
[51] Zhuyun Dai and Jamie Callan. 2020. Context-aware term weighting for first stage passage retrieval. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'20)*. 1533–1536.
[52] Zhuyun Dai, Vincent Zhao, Ji Ma, Yi Luan, Jianmo Ni, Jing Lu, Anton Bakalov, Kelvin Guu, Keith B. Hall, and Ming-Wei Chang. 2022. Promptagator: Few-shot dense retrieval from 8 examples. Retrieved from https://abs/2209.11755
[53] 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 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 4171–4186.
[54] Thomas Diggelmann, Jordan Boyd-Graber, Jannis Bulian, Massimiliano Ciaramita, and Markus Leippold. 2020. CLIMATE-FEVER: A dataset for verification of real-world climate claims. Retrieved from https://arXiv:2012.00614
[55] Bhargav Dodla, Akash Kumar Mohankumar, and Amit Singh. 2022. HEARTS: Multi-task fusion of dense retrieval and non-autoregressive generation for sponsored search. Retrieved from https://abs/2209.05861
[56] Miao Fan, Jiacheng Guo, Shuai Zhu, Shuo Miao, Mingming Sun, and Ping Li. 2019. MOBIUS: Towards the next generation of query-ad matching in baidu's sponsored search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'19)*. 2509–2517.
[57] Yixing Fan, Xiaohui Xie, Yinqiong Cai, Jia Chen, Xinyu Ma, Xiangsheng Li, Ruqing Zhang, Jiafeng Guo, and Yiqun Liu. 2021. Pre-training methods in information retrieval. Retrieved from https://arXiv:2111.13853
[58] Hui Fang, Tao Tao, and ChengXiang Zhai. 2004. A formal study of information retrieval heuristics. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'04)*.
[59] Hui Fang and ChengXiang Zhai. 2005. An exploration of axiomatic approaches to information retrieval. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'05)*.
[60] Hui Fang and ChengXiang Zhai. 2006. Semantic term matching in axiomatic approaches to information retrieval. *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.
[61] William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *J. Mach. Learn. Res.* 23, 1 (2022), 5232–5270.

[62] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE v2: Sparse lexical and expansion model for information retrieval. Retrieved from https://arXiv:2109.10086
[63] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse lexical and expansion model for first stage ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2288–2292.
[64] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. A white box analysis of ColBERT. In *Proceedings of the European Conference on Information Retrieveal (ECIR'21)*.
[65] Luyu Gao and Jamie Callan. 2021. Condenser: A pre-training architecture for dense retrieval. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 981–993.
[66] Luyu Gao and Jamie Callan. 2021. Unsupervised corpus aware language model pre-training for dense passage retrieval. Retrieved from https://arXiv:2108.05540
[67] Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021. COIL: Revisit exact lexical match in information retrieval with contextualized inverted list. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 3030–3042.
[68] Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021. Rethink training of BERT rerankers in multi-stage retrieval pipeline. In *Proceedings of the Advances in Information Retrieval 43rd European Conference on IR Research (ECIR'21)*, Vol. 12657. 280–286.
[69] Luyu Gao, Xueguang Ma, Jimmy J. Lin, and Jamie Callan. 2022. Tevatron: An efficient and flexible toolkit for dense retrieval. Retrieved from https://abs/2203.05765
[70] Luyu Gao, Yunyi Zhang, Jiawei Han, and Jamie Callan. 2021. Scaling deep contrastive learning batch size under memory limited setup. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP'21)*. 316–321.
[71] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'21)*.
[72] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. 2013. Optimized product quantization for approximate nearest-neighbor search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'13)*. 2946–2953.
[73] Daniel Gillick, Sayali Kulkarni, Larry Lansing, Alessandro Presta, Jason Baldridge, Eugene Ie, and Diego Garcia-Olano. 2019. Learning dense representations for entity retrieval. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL'19)*. 528–537.
[74] Aristides Gionis, Piotr Indyk, Rajeev Motwani et al. 1999. Similarity search in high dimensions via hashing. In *Proceedings of the International Conference on Very Large Data Bases (VLDB'99)*, Vol. 99. 518–529.
[75] 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 (CIKM'16)*. 55–64.
[76] Jiafeng Guo, Yixing Fan, Xiang Ji, and Xueqi Cheng. 2019. MatchZoo: A learning, practicing, and developing system for neural text matching. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19)*. 1297–1300.
[77] Jiafeng Guo, Yixing Fan, Liang Pang, Liu Yang, Qingyao Ai, Hamed Zamani, Chen Wu, W. Bruce Croft, and Xueqi Cheng. 2020. A deep look into neural ranking models for information retrieval. *Info. Process. Manage.* 57, 6 (2020), 102067.
[78] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. 2020. Accelerating large-scale inference with anisotropic vector quantization. In *Proceedings of the 37th International Conference on Machine Learning (ICML'20) (Proceedings of Machine Learning Research, Vol. 119)*. 3887–3896.
[79] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: Retrieval-augmented language model pre-training. Retrieved from https://abs/2002.08909
[80] Kiana Hajebi, Yasin Abbasi-Yadkori, Hossein Shahbazi, and Hong Zhang. 2011. Fast approximate nearest-neighbor search with k-nearest-neighbor graph. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI'11)*. 1312–1317.
[81] Donna Harman. 2011. Information retrieval evaluation. *Synth. Lect. Info. Concepts, Retriev. Serv.* 3, 2 (2011), 1–119.
[82] 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.
[83] Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. 2021. Efficient nearest-neighbor language models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'21)*.
[84] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'20)*. 9726–9735.

[85] Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun-Hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply. Retrieved from https://arXiv:1705.00652

[86] Jonathan Herzig, Thomas Müller, Syrine Krichene, and Julian Eisenschlos. 2021. Open domain question answering over tables via dense retrieval. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 512–519.

[87] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. Retrieved from https://abs/1503.02531

[88] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.* 9, 8 (1997), 1735–1780.

[89] Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. 2020. Improving efficient neural ranking models with cross-architecture knowledge distillation. Retrieved from https://arXiv:2010.02666

[90] Sebastian Hofstatter, Nick Craswell, Bhaskar Mitra, Hamed Zamani, and Allan Hanbury. 2022. Are we there yet? A decision framework for replacing term based retrieval with dense retrieval systems. Retrieved from https://abs/2206.12993

[91] Sebastian Hofstätter, O. Khattab, Sophia Althammer, Mete Sertkan, and Allan Hanbury. 2022. Introducing neural bag of whole-words with colberter: Contextualized late interactions using enhanced reduction. Retrieved from https://abs/2203.13088

[92] 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.

[93] Wu Hong, Zhuosheng Zhang, Jinyuan Wang, and Hai Zhao. 2022. Sentence-aware contrastive learning for open-domain passage retrieval. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. 1062–1074.

[94] Doris Hoogeveen, Karin M. Verspoor, and Timothy Baldwin. 2015. CQADupStack: A benchmark data set for community question-answering research. In *Proceedings of the 20th Australasian Document Computing Symposium*. 1–8.

[95] Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. 2020. Embedding-based retrieval in facebook search. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'20)*. 2553–2561.

[96] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry P. Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management (CIKM'13)*. 2333–2338.

[97] Patrick Huber, Armen Aghajanyan, Barlas Oğuz, Dmytro Okhonko, Wen-tau Yih, Sonal Gupta, and Xilun Chen. 2021. CCQA: A new web-scale question answering dataset for model pre-training. Retrieved from https://arXiv:2110.07731

[98] Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2020. Poly-encoders: Architectures and pre-training strategies for fast and accurate multi-sentence scoring. In *Proceedings of the 8th International Conference on Learning Representations (ICLR'20)*.

[99] Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC'98)*.

[100] Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. [n. d.]. First Quora Dataset Release: Question Pairs. https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

[101] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Towards unsupervised dense information retrieval with contrastive learning. Retrieved from https://abs/2112.09118

[102] Gautier Izacard and Edouard Grave. 2021. Distilling knowledge from reader to retriever for question answering. In *Proceedings of the International Conference on Learning Representations*.

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

[104] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.* 20, 4 (2002), 422–446.

[105] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. 2011. Product quantization for nearest-neighbor search. *IEEE Trans. Pattern Anal. Mach. Intell.* 33, 1 (2011), 117–128.

[106] Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C. Park. 2022. Augmenting document representations for dense retrieval with interpolation and perturbation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. 442–452.

[107] Zongcheng Ji, Zhengdong Lu, and Hang Li. 2014. An information retrieval approach to short text conversation. Retrieved from https://abs/1408.6988

[108] J. Johnson, M. Douze, and H. Jegou. 2019. Billion-scale similarity search with GPUs. *IEEE Trans. Big Data* 7, 3 (2019), 535–547.

[109] Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. 1601–1611.

[110] T. Joyce and R. M. Needham. 1958. The thesaurus approach to information retrieval. *American Document.* 9, 3 (1958), 192.

[111] 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 Conference on Empirical Methods in Natural Language Processing (EMNLP'20)*. 6769–6781.

[112] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models. In *Proceedings of the 8th International Conference on Learning Representations (ICLR'20)*.

[113] 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 (SIGIR'20)*. 39–48.

[114] Jon M. Kleinberg. 2000. The small-world phenomenon: An algorithmic perspective. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*. 163–170.

[115] Bogdan Kostić, Julian Risch, and Timo Möller. 2021. Multi-modal retrieval of tables and texts using tri-encoder models. In *Proceedings of the 3rd Workshop on Machine Reading for Question Answering*. 82–91.

[116] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Trans. Assoc. Comput. Linguist.* 7 (2019), 452–466.

[117] Hyunji Lee, Sohee Yang, Hanseok Oh, and Minjoon Seo. 2022. Generative retrieval for long sequences. Retrieved from https://abs/2204.13596

[118] Jinhyuk Lee, Mujeen Sung, Jaewoo Kang, and Danqi Chen. 2021. Learning dense representations of phrases at scale. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*. 6634–6647.

[119] Jinhyuk Lee, Alexander Wettig, and Danqi Chen. 2021. Phrase retrieval learns passage retrieval, too. Retrieved from https://arXiv:2109.08133

[120] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 6086–6096.

[121] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. 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*. 7871–7880.

[122] Patrick Lewis, Barlas Oğuz, Wenhan Xiong, Fabio Petroni, Wen tau Yih, and Sebastian Riedel. 2021. Boosted dense retriever. Retrieved from https://abs/2112.07771

[123] Patrick Lewis, Yuxiang Wu, Linqing Liu, Pasquale Minervini, Heinrich Küttler, Aleksandra Piktus, Pontus Stenetorp, and Sebastian Riedel. 2021. Paq: 65 million probably-asked questions and what you can do with them. Retrieved from https://arXiv:2102.07033

[124] Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of the Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems (NeurIPS'20)*.

[125] Hang Li. 2011. Learning to rank for information retrieval and natural language processing. *Synth. Lect. Hum. Lang. Technol.* 4 (2011), 1–113.

[126] Hang Li, Ahmed Mourad, Shengyao Zhuang, Bevan Koopman, and Guido Zuccon. 2021. Pseudo relevance feedback with deep language models and dense retrievers: Successes and pitfalls. Retrieved from https://arXiv:2108.11044

[127] Hang Li, Shengyao Zhuang, Ahmed Mourad, Xueguang Ma, Jimmy J. Lin, and G. Zuccon. 2021. Improving query representations for dense retrieval with pseudo relevance feedback: A reproducibility study. Retrieved from https://abs/2112.06400

[128] Jiwei Li, Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. 1106–1115.

[129] Minghan Li, Ming Li, Kun Xiong, and Jimmy Lin. 2021. Multi-task dense retrieval via model uncertainty fusion for open-domain question answering. In *Findings of the Association for Computational Linguistics (EMNLP'21)*. 274–287.
[130] Sen Li, Fuyu Lv, Taiwei Jin, Guli Lin, Keping Yang, Xiaoyi Zeng, Xiao-Ming Wu, and Qianli Ma. 2021. Embedding-based product retrieval in Taobao search. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3181–3189.
[131] Wen Li, Ying Zhang, Yifang Sun, Wei Wang, Mingjie Li, Wenjie Zhang, and Xuemin Lin. 2019. Approximate nearest-neighbor search on high dimensional data—Experiments, analyses, and improvement. *IEEE Trans. Knowl. Data Eng.* 32, 8 (2019), 1475–1488.
[132] Yizhi Li, Zhenghao Liu, Chenyan Xiong, and Zhiyuan Liu. 2021. More robust dense retrieval with contrastive dual learning. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*. 287–296.
[133] Yongqi Li, Nan Yang, Liang Wang, Furu Wei, and Wenjie Li. 2023. Multiview identifiers enhanced generative retrieval. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 6636–6648.
[134] Zehan Li, Nan Yang, Liang Wang, and Furu Wei. 2022. Learning diverse document representations with deep query interactions for dense retrieval. Retrieved from https://arXiv:2208.04232
[135] Davis Liang, Peng Xu, Siamak Shakeri, Cicero Nogueira dos Santos, Ramesh Nallapati, Zhiheng Huang, and Bing Xiang. 2020. Embedding-based zero-shot retrieval through query generation. Retrieved from https://arXiv:2009.10270
[136] 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 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'21)*. 2356–2362.
[137] Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2021. Pretrained transformers for text ranking: Bert and beyond. *Synth. Lect. Hum. Lang. Technol.* 14, 4 (2021), 1–325.
[138] Jimmy J. Lin and Xueguang Ma. 2021. A few brief notes on deepimpact, COIL, and a conceptual framework for information retrieval techniques. Retrieved from https://abs/2106.14807
[139] Sheng-Chieh Lin, Minghan Li, and Jimmy Lin. 2022. Aggretriever: A simple approach to aggregate textual representation for robust dense passage retrieval. Retrieved from https://abs/2208.00511
[140] Sheng-Chieh Lin and Jimmy J. Lin. 2021. Densifying sparse representations for passage retrieval by representational slicing. Retrieved from https://abs/2112.04666
[141] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2020. Distilling dense representations for ranking using tightly-coupled teachers. Retrieved from https://arXiv:2010.11386
[142] Zhenghao Lin, Yeyun Gong, Xiao Liu, Hang Zhang, Chen Lin, Anlei Dong, Jian Jiao, Jingwen Lu, Daxin Jiang, Rangan Majumder, et al. 2023. PROD: Progressive distillation for dense retrieval. In *Proceedings of the ACM Web Conference 2023*. 3299–3308.
[143] Alexander Liu and Samuel Yang. 2022. Masked autoencoders as the unified learners for pre-trained sentence representation. Retrieved from https://abs/2208.00231
[144] Fangyu Liu, Serhii Havrylov, Yunlong Jiao, Jordan Massiah, and Emine Yilmaz. 2021. Trans-encoder: Unsupervised sentence-pair modelling through self-and mutual-distillations. Retrieved from https://arXiv:2109.13059
[145] Jiawei Liu, Yangyang Kang, Di Tang, Kaisong Song, Changlong Sun, Xiaofeng Wang, Wei Lu, and Xiaozhong Liu. 2022. Order-disorder: Imitation adversarial attacks for black-box neural ranking models. Retrieved from https://abs/2209.06506
[146] Jiduan Liu, Jiahao Liu, Yang Yang, Jingang Wang, Wei Wu, Dongyan Zhao, and Rui Yan. 2022. GNN-encoder: Learning a dual-encoder architecture via graph neural networks for passage retrieval. Retrieved from https://arXiv:2204.08241
[147] Linqing Liu, Patrick Lewis, Sebastian Riedel, and Pontus Stenetorp. 2021. Challenges in generalization in open domain question answering. Retrieved from https://abs/2109.01156
[148] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surveys*, 55, 9 (2023), 1–35.
[149] Tie-Yan Liu. 2010. Learning to rank for information retrieval. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'10)*. 904.
[150] Ye Liu, Kazuma Hashimoto, Yingbo Zhou, Semih Yavuz, Caiming Xiong, and S Yu Philip. 2021. Dense hierarchical retrieval for open-domain question answering. In *Proceedings of the Association for Computational Linguistics (EMNLP'21)*. 188–200.
[151] Yiding Liu, Guan Huang, Jiaxiang Liu, Weixue Lu, Suqi Cheng, Yukun Li, Daiting Shi, Shuaiqiang Wang, Zhicong Cheng, and Dawei Yin. 2021. Pre-trained language model for web-scale retrieval in baidu search. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

[152] 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. Retrieved from https://arXiv:1907.11692

[153] Yiqun Liu, Kaushik Rangadurai, Yunzhong He, Siddarth Malreddy, Xunlong Gui, Xiaoyi Liu, and Fedor Borisyuk. 2021. Que2Search: Fast and accurate query and document understanding for search at Facebook. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

[154] Zheng Liu and Yingxia Shao. 2022. RetroMAE: Pre-training retrieval-oriented transformers via masked auto-encoder. Retrieved from https://abs/2205.12035

[155] Zhenghao Liu, Kaitao Zhang, Chenyan Xiong, Zhiyuan Liu, and Maosong Sun. 2021. OpenMatch: An open source library for Neu-IR research. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'21)*. 2531–2535.

[156] Jing Lu, Gustavo Hernández Ábrego, Ji Ma, Jianmo Ni, and Yinfei Yang. 2020. Neural passage retrieval with improved negative contrast. Retrieved from https://abs/2010.12523

[157] Jing Lu, Gustavo Hernández Ábrego, Ji Ma, Jianmo Ni, and Yinfei Yang. 2021. Multi-stage training with improved negative contrast for neural passage retrieval. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 6091–6103.

[158] Shuai Lu, Nan Duan, Hojae Han, Daya Guo, Seung-won Hwang, and Alexey Svyatkovskiy. 2022. ReACC: A retrieval-augmented code completion framework. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL'22)*. 6227–6240.

[159] Shuqi Lu, Di He, Chenyan Xiong, Guolin Ke, Waleed Malik, Zhicheng Dou, Paul Bennett, Tie-Yan Liu, and Arnold Overwijk. 2021. Less is more: Pretrain a strong siamese encoder for dense text retrieval using a weak decoder. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'21)*. 2780–2791.

[160] Yuxiang Lu, Yiding Liu, Jiaxiang Liu, Yunsheng Shi, Zhengjie Huang, Shikun Feng, Yu Sun, Hao Tian, Hua Wu, Shuaiqiang Wang, Dawei Yin, and Haifeng Wang. 2022. ERNIE-search: Bridging cross-encoder with dual-encoder via self on-the-fly distillation for dense passage retrieval. Retrieved from https://abs/2205.09153

[161] Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. Sparse, dense, and attentional representations for text retrieval. *Trans. Assoc. Comput. Linguist.* 9 (2021), 329–345.

[162] Man Luo, Arindam Mitra, Tejas Gokhale, and Chitta Baral. 2022. Improving biomedical information retrieval with neural retrievers. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI'22), 34th Conference on Innovative Applications of Artificial Intelligence (IAAI'22), and the 12th Symposium on Educational Advances in Artificial Intelligence (EAAI'22)*. 11038–11046.

[163] Ji Ma, Ivan Korotkov, Yinfei Yang, Keith Hall, and Ryan McDonald. 2021. Zero-shot neural passage retrieval via domain-targeted synthetic question generation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*. 1075–1088.

[164] Xinyu Ma, Jiafeng Guo, Ruqing Zhang, Yixing Fan, and Xueqi Cheng. 2022. Pre-train a discriminative text encoder for dense retrieval via contrastive span prediction. In *Proceedings of the 45rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 848–858.

[165] Xinyu Ma, Jiafeng Guo, Ruqing Zhang, Yixing Fan, Xiang Ji, and Xueqi Cheng. 2021. PROP: Pre-training with representative words prediction for ad hoc retrieval. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 283–291.

[166] 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 (SIGIR'21)*. 1318–1327.

[167] Xinyu Ma, Ruqing Zhang, Jiafeng Guo, Yixing Fan, and Xueqi Cheng. 2022. A contrastive pre-training approach to learn discriminative autoencoder for dense retrieval. Retrieved from https://arXiv:2208.09846

[168] Zhengyi Ma, Zhicheng Dou, Wei Xu, Xinyu Zhang, Hao Jiang, Zhao Cao, and Ji-Rong Wen. 2021. Pre-training for ad hoc retrieval: Hyperlink is also you need. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*. 1212–1221.

[169] Sean MacAvaney, Sergey Feldman, Nazli Goharian, Doug Downey, and Arman Cohan. 2020. ABNIRML: Analyzing the behavior of neural IR models. Retrieved from https://arXiv:2011.00696

[170] Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. 2019. CEDR: Contextualized embeddings for document ranking. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19)*. 1101–1104.

[171] Alessandro Magnani, Feng Liu, Suthee Chaidaroon, Sachin Yadav, Praveen Reddy Suram, Ajit Puthenputhussery, Sijie Chen, Min Xie, Anirudh Kashi, Tony Lee, and Ciya Liao. 2022. Semantic retrieval at walmart. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'22)*. 3495–3503.

[172] Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. 2018. WWW'18 open challenge: Financial opinion mining and question answering. In *Proceedings of the Web Conference (WWW'18)*. 1941–1942.

[173] Jean Maillard, Vladimir Karpukhin, Fabio Petroni, Wen-tau Yih, Barlas Oğuz, Veselin Stoyanov, and Gargi Ghosh. 2021. Multi-task retrieval for knowledge-intensive tasks. Retrieved from https://arXiv:2101.00117

[174] Yury Malkov, Alexander Ponomarenko, Andrey Logvinov, and Vladimir Krylov. 2014. Approximate nearest-neighbor algorithm based on navigable small world graphs. *Inf. Syst.* 45 (2014), 61–68.

[175] Yu A. Malkov and Dmitry A. Yashunin. 2018. Efficient and robust approximate nearest-neighbor search using hierarchical navigable small world graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* 42, 4 (2018), 824–836.

[176] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK.

[177] Kelong Mao, Zhicheng Dou, Haonan Chen, Fengran Mo, and Hongjin Qian. 2023. Large language models know your contextual search intent: A prompting framework for conversational search. Retrieved from https://arXiv:2303.06573

[178] Melvin Earl Maron and John Larry Kuhns. 1960. On relevance, probabilistic indexing and information retrieval. *J. ACM* 7, 3 (1960), 216–244.

[179] Ken J. McDonell. 1977. An inverted index implementation. *Comput. J.* 20, 2 (1977), 116–123.

[180] Donald Metzler, Yi Tay, Dara Bahri, and Marc Najork. 2021. Rethinking search: Making experts out of dilettantes. Retrieved from https://abs/2105.02274

[181] Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems (NeurIPS'13)*. 3111–3119.

[182] Bhaskar Mitra and Nick Craswell. 2017. Neural models for information retrieval. Retrieved from https://arXiv:1705.01509

[183] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web (WWW'17)*. 1291–1299.

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

[185] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches Co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS'16)*.

[186] Jianmo Ni, Gustavo Hern'andez 'Abrego, Noah Constant, Ji Ma, Keith B. Hall, Daniel Matthew Cer, and Yinfei Yang. 2021. Sentence-T5: Scalable sentence encoders from pre-trained text-to-text models. Retrieved from https://abs/2108.08877

[187] Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hern'andez 'Abrego, Ji Ma, Vincent Zhao, Yi Luan, Keith Hall, Ming-Wei Chang, and Yinfei Yang. 2021. Large dual encoders are generalizable retrievers. Retrieved from https://abs/2112.07899

[188] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with BERT. Retrieved from https://abs/1901.04085

[189] Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document ranking with a pretrained sequence-to-sequence model. In *Proceedings of the Association for Computational Linguistics (EMNLP'20)*. 708–718.

[190] Rodrigo Nogueira, Jimmy Lin, and AI Epistemic. 2019. From doc2query to docTTTTTquery. *Online Preprint* 6 (2019), 2.

[191] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-stage document ranking with BERT. Retrieved from https://abs/1910.14424

[192] Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Scott Yih. 2020. Unik-qa: Unified representations of structured and unstructured knowledge for open-domain question answering. Retrieved from https://arXiv:2012.14610

[193] Barlas Oğuz, Kushal Lakhotia, Anchit Gupta, Patrick Lewis, Vladimir Karpukhin, Aleksandra Piktus, Xilun Chen, Sebastian Riedel, Wen-tau Yih, Sonal Gupta et al. 2021. Domain-matched pre-training tasks for dense retrieval. Retrieved from https://arXiv:2107.13602

[194] Liang Pang, Jun Xu, Qingyao Ai, Yanyan Lan, Xueqi Cheng, and Jirong Wen. 2020. SetRank: Learning a permutation-invariant ranking model for information retrieval. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval.*

[195] Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen et al. 2023. Check your facts and try again: Improving large language models with external knowledge and automated feedback. Retrieved from https://arXiv:2302.12813

[196] Gustavo Penha, Arthur Câmara, and Claudia Hauff. 2022. Evaluating the robustness of retrieval pipelines with query variation generators. In *Advances in Information Retrieval 44th European Conference on IR Research (ECIR'22)*. 397–412.

[197] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2227–2237.

[198] 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 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2523–2544.

[199] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP'19)*. 2463–2473.

[200] Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Dmytro Okhonko, Samuel Broscheit, Gautier Izacard, Patrick Lewis, Barlas Oguz, Edouard Grave, Wen-tau Yih, and Sebastian Riedel. 2021. The web is your oyster—Knowledge-intensive NLP against a very large web corpus. Retrieved from https://abs/2112.09924

[201] Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 4996–5001.

[202] Prafull Prakash, Julian Killingback, and Hamed Zamani. 2021. Learning robust dense retrieval models from incomplete relevance labels. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1728–1732.

[203] Yifan Qiao, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2019. Understanding the behaviors of BERT in ranking. Retrieved from https://abs/1904.07531

[204] Yifan Qiao, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2019. Understanding the behaviors of BERT in ranking. Retrieved from https://abs/1904.07531

[205] Tao Qin, Wei Chen, and Tie-Yan Liu. 2014. Sponsored search auctions: Recent advances and future directions. *ACM Trans. Intell. Syst. Technol.* 5, 4 (2014), 60:1–60:34.

[206] Yifu Qiu, Hongyu Li, Yingqi Qu, Ying Chen, Qiaoqiao She, Jing Liu, Hua Wu, and Haifeng Wang. 2022. DuReader_retrieval: A large-scale chinese benchmark for passage retrieval from web search engine. Retrieved from https://arXiv:2203.10232

[207] 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 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 5835–5847.

[208] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning (ICML'21)*. 8748–8763.

[209] Alec Radford and Karthik arasimhan. 2018. Improving language understanding by generative pre-training. Online preprint (2018).

[210] Colin Raffel, Noam M. 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. Retrieved from https://abs/1910.10683

[211] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 2383–2392.

[212] Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. Retrieved from https://arXiv:2302.00083

[213] Ori Ram, Gal Shachaf, Omer Levy, Jonathan Berant, and Amir Globerson. 2021. Learning to retrieve passages without supervision. Retrieved from https://abs/2112.07708

[214] Revanth Gangi Reddy, Vikas Yadav, Md Arafat Sultan, Martin Franz, Vittorio Castelli, Heng Ji, and Avirup Sil. 2021. Towards robust neural retrieval models with synthetic pre-training. Retrieved from https://arXiv:2104.07800

[215] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP'19)*. 3982–3992.

[216] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP'19)*. 3982–3992.

[217] Nils Reimers and Iryna Gurevych. 2021. The curse of dense low-dimensional information retrieval for large index sizes. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*. 605–611.

[218] Ruiyang Ren, Shangwen Lv, Yingqi Qu, Jing Liu, Wayne Xin Zhao, QiaoQiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. 2021. PAIR: Leveraging passage-centric similarity relation for improving dense passage retrieval. In *Proceedings of the Association for Computational Linguistics (ACL-IJCNLP'21)*. 2173–2183.

[219] Ruiyang Ren, Yingqi Qu, Jing Liu, Wayne Xin Zhao, Qiaoqiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. 2021. RocketQAv2: A joint training method for dense passage retrieval and passage re-ranking. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 2825–2835.

[220] Ruiyang Ren, Yingqi Qu, Jing Liu, Wayne Xin Zhao, Qifei Wu, Yuchen Ding, Hua Wu, Haifeng Wang, and Ji-Rong Wen. 2023. A thorough examination on zero-shot dense retrieval. In *Proceedings of the Association for Computational Linguistics (EMNLP'23)*.

[221] Ruiyang Ren, Yuhao Wang, Yingqi Qu, Wayne Xin Zhao, Jing Liu, Hao Tian, Hua Wu, Ji-Rong Wen, and Haifeng Wang. 2023. Investigating the factual knowledge boundary of large language models with retrieval augmentation. Retrieved from https://arXiv:2307.11019

[222] Ruiyang Ren, Wayne Xin Zhao, Jing Liu, Hua Wu, Ji-Rong Wen, and Haifeng Wang. 2023. TOME: A two-stage approach for model-based retrieval. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 6102–6114.

[223] Mehdi Rezagholizadeh, Aref Jafari, Puneeth S. M. Saladi, Pranav Sharma, Ali Saheb Pasand, and Ali Ghodsi. 2022. Pro-KD: Progressive distillation by following the footsteps of the teacher. In *Proceedings of the 29th International Conference on Computational Linguistics (COLING'22)*. 4714–4727.

[224] Kirk Roberts, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, Kyle Lo, Ian Soboroff, Ellen Voorhees, Lucy Lu Wang, and William R. Hersh. 2020. TREC-COVID: Rationale and structure of an information retrieval shared task for COVID-19. *J. Amer. Med. Info. Assoc.* 27, 9, 1431–1436.

[225] Stephen Robertson. 2004. Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of documentation* 60, 5 (2004), 503–520.

[226] Stephen Robertson and Hugo Zaragoza. 2009. *The Probabilistic Relevance Framework: BM25 and Beyond*. Now Publishers Inc., Hanover, MD.

[227] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline M. Hancock-Beaulieu, Mike Gatford et al. 1995. Okapi at TREC-3. *Nist Special Publ.* 109 (1995), 109.

[228] Corby Rosset, Bhaskar Mitra, Chenyan Xiong, Nick Craswell, Xia Song, and Saurabh Tiwary. 2019. An axiomatic approach to regularizing neural ranking models. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19)*, Benjamin Piwowarski, Max Chevalier, Éric Gaussier, Yoelle Maarek, Jian-Yun Nie, and Falk Scholer (Eds.). ACM, 981–984.

[229] Devendra Sachan, Mostofa Patwary, Mohammad Shoeybi, Neel Kant, Wei Ping, William L. Hamilton, and Bryan Catanzaro. 2021. End-to-end training of neural retrievers for open-domain question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*. 6648–6662.

[230] Devendra Singh Sachan, Mike Lewis, Dani Yogatama, Luke Zettlemoyer, Joelle Pineau, and Manzil Zaheer. 2022. Questions are all you need to train a dense passage retriever. Retrieved from https://arXiv:2206.10658

[231] Gerard Salton. 1962. Some experiments in the generation of word and document associations. In *Proceedings of the Computer Conference of the American Federation of Information Processing Societies (AFIPS'62)*. 234–250.

[232] Gerard Salton and Chris Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manag.* 24 (1988), 513–523.

[233] Gerard Salton, Anita Wong, and Chung-Shu Yang. 1975. A vector space model for automatic indexing. *Commun. ACM* 18, 11 (1975), 613–620.

[234] Mark Sanderson et al. 2010. Test collection based evaluation of information retrieval systems. *Foundations and Trends in Information Retrieval* 4, 4 (2010), 247–375.

[235] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2021. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. Retrieved from https://arXiv:2112.01488

[236] Christopher Sciavolino, Zexuan Zhong, Jinhyuk Lee, and Danqi Chen. 2021. Simple entity-centric questions challenge dense retrievers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'21)*.

[237] Minjoon Seo, Tom Kwiatkowski, Ankur Parikh, Ali Farhadi, and Hannaneh Hajishirzi. 2018. Phrase-indexed question answering: A new challenge for scalable document comprehension. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 559–564.

[238] Minjoon Seo, Jinhyuk Lee, Tom Kwiatkowski, Ankur Parikh, Ali Farhadi, and Hannaneh Hajishirzi. 2019. Real-time open-domain question answering with dense-sparse phrase index. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 4430–4441.

[239] Siamak Shakeri, Cicero Nogueira dos Santos, Henghui Zhu, Patrick Ng, Feng Nan, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang. 2020. End-to-end synthetic data generation for domain adaptation of question answering systems. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'20)*. 5445–5460.

[240] Tao Shen, Xiubo Geng, Chongyang Tao, Can Xu, Xiaolong Huang, Binxing Jiao, Linjun Yang, and Daxin Jiang. 2022. LexMAE: Lexicon-bottlenecked pretraining for large-scale retrieval. Retrieved from https://abs/2208.14754

[241] Xiaoyu Shen, Svitlana Vakulenko, Marco Del Tredici, Gianni Barlacchi, Bill Byrne, and Adrià de Gispert. 2022. Low-resource dense retrieval for open-domain question answering: A comprehensive survey. Retrieved from https://abs/2208.03197

[242] Anshumali Shrivastava and Ping Li. 2014. Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS). In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems*. 2321–2329.

[243] Georgios Sidiropoulos and Evangelos Kanoulas. 2022. Analysing the robustness of dual encoders for dense retrieval against misspellings. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'22)*. 2132–2136.

[244] Harsha Vardhan Simhadri, G. R. Williams, Martin Aumüller, Matthijs Douze, Artem Babenko, Dmitry Baranchuk, Qi Chen, Lucas Hosseini, Ravishankar Krishnaswamy, Gopal Srinivasa, Suhas Jayaram Subramanya, and Jingdong Wang. 2022. Results of the NeurIPS'21 challenge on billion-scale approximate nearest-neighbor search. Retrieved from https://abs/2205.03763

[245] Josef Sivic and Andrew Zisserman. 2003. Video google: A text-retrieval approach to object matching in videos. *Proceedings of the 9th IEEE International Conference on Computer Vision*. 1470–1477.

[246] Ian Soboroff, Shudong Huang, and Donna Harman. 2019. TREC 2019 news track overview. In *Proceedings of the Text Retrieval Conference (TREC'19)*.

[247] Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of the Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems*. 801–809.

[248] Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 151–161.

[249] Axel Suarez, Dyaa Albakour, David Corney, Miguel Martinez, and José Esquivel. 2018. A data collection for evaluating the retrieval of related tweets to news articles. In *Proceedings of the European Conference on Information Retrieval*. Springer, 780–786.

[250] Suhas Jayaram Subramanya, Fnu Devvrit, Harsha Vardhan Simhadri, Ravishankar Krishnaswamy, and Rohan Kadekodi. 2019. DiskANN: Fast accurate billion-point nearest-neighbor search on a single node. In *Proceedings of the Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems (NeurIPS'19)*.

[251] 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. Retrieved from https://arXiv:2304.09542

[252] Yutao Sun, Li Dong, Barun Patra, Shuming Ma, Shaohan Huang, Alon Benhaim, Vishrav Chaudhary, Xia Song, and Furu Wei. 2022. A length-extrapolatable transformer. Retrieved from https://arXiv:2212.10554

[253] Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. ERNIE 2.0: A continual pre-training framework for language understanding. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI'20), the 32nd Innovative Applications of Artificial Intelligence Conference (IAAI'20), the 10th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI'20)*. 8968–8975.

[254] Weng Lam Tam, Xiao Liu, Kaixuan Ji, Lilong Xue, Xing Zhang, Yuxiao Dong, Jiahua Liu, Maodi Hu, and Jie Tang. 2022. Parameter-efficient prompt tuning makes generalized and calibrated neural text retrievers. Retrieved from https://abs/2207.07087

[255] Alexandre Tamborrino. [n. d.]. Introducing Natural Language Search for Podcast Episodes. https://engineering.atspotify.com/2022/03/introducing-natural-language-search-for-podcast-episodes/.

[256] Hongyin Tang, Xingwu Sun, Beihong Jin, Jingang Wang, Fuzheng Zhang, and Wei Wu. 2021. Improving document representations by generating pseudo query embeddings for dense retrieval. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*. 5054–5064.

[257] Zhengyang Tang, Benyou Wang, and Ting Yao. 2022. DPTDR: Deep prompt tuning for dense passage retrieval. Retrieved from https://arXiv:2208.11503

[258] Yi Tay, Vinh Quang Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, Tal Schuster, William W. Cohen, and Donald Metzler. 2022. Transformer memory as a differentiable search index. Retrieved from https://abs/2202.06991

[259] Nandan Thakur, Nils Reimers, Johannes Daxenberger, and Iryna Gurevych. 2021. Augmented SBERT: Data augmentation method for improving bi-encoders for pairwise sentence scoring tasks. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 296–310.

[260] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A heterogenous benchmark for zero-shot evaluation of information retrieval models. Retrieved from https://arXiv:2104.08663

[261] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: A large-scale dataset for fact extraction and VERification. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 809–819.

[262] Naftali Tishby and Noga Zaslavsky. 2015. Deep learning and the information bottleneck principle. In *Proceedings of the IEEE Information Theory Workshop (ITW'15)*. 1–5.

[263] Nicola Tonellotto. 2022. Lecture notes on neural information retrieval. Retrieved from https://abs/2207.13443

[264] Nicola Tonellotto and Craig Macdonald. 2021. Query embedding pruning for dense retrieval. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*. 3453–3457.

[265] George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R. Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos et al. 2015. An overview of the BIOASQ large-scale biomedical semantic indexing and question answering competition. *BMC Bioinform.* 16, 1 (2015), 1–28.

[266] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. Retrieved from https://abs/1807.03748 (2018).

[267] Saúl Vargas. 2014. Novelty and diversity enhancement and evaluation in recommender systems and information retrieval. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'14)*. 1281.

[268] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*. 5998–6008.

[269] Michael Völske, Alexander Bondarenko, Maik Fröbe, Benno Stein, Jaspreet Singh, Matthias Hagen, and Avishek Anand. 2021. Towards axiomatic explanations for neural ranking models. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*. 13–22.

[270] Ellen M. Voorhees et al. 1999. The trec-8 question answering track report. In *Proceedings of the Text Retrieval Conference (TREC'99)*, Vol. 99. 77–82.

[271] 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*. 241–251.

[272] David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. 2020. Fact or fiction: Verifying scientific claims. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'20)*. 7534–7550.

[273] Jiexin Wang, Adam Jatowt, and Masatoshi Yoshikawa. 2021. ArchivalQA: A large-scale benchmark dataset for open domain question answering over archival news collections. Retrieved from https://arXiv:2109.03438

[274] Jingdong Wang and Shipeng Li. 2012. Query-driven iterated neighborhood graph search for large scale indexing. *Proceedings of the 20th ACM International Conference on Multimedia*.

[275] Jingdong Wang, Ting Zhang, Jingkuan Song, N. Sebe, and Heng Tao Shen. 2018. A survey on learning to hash. *IEEE Trans. Pattern Anal. Mach. Intell.* 40 (2018), 769–790.

[276] Kexin Wang, Nils Reimers, and Iryna Gurevych. 2021. TSDAE: Using transformer-based sequential denoising auto-encoder for unsupervised sentence embedding learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'21)*.

[277] Kexin Wang, Nandan Thakur, Nils Reimers, and Iryna Gurevych. 2021. GPL: Generative pseudo labeling for unsupervised domain adaptation of dense retrieval. Retrieved from https://arXiv:2112.07577

[278] Kaiye Wang, Qiyue Yin, Wei Wang, Shu Wu, and Liang Wang. 2016. A comprehensive survey on cross-modal retrieval. Retrieved from https://abs/1607.06215 (2016).

[279] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. SimLM: Pre-training with representation bottleneck for dense passage retrieval. Retrieved from https://abs/2207.02578.

[280] Xiao Wang, Craig Macdonald, Nicola Tonellotto, and Iadh Ounis. 2021. Pseudo-relevance feedback for multiple representation dense retrieval. Retrieved from https://arXiv:2106.11251

[281] Xiao Wang, Craig Macdonald, Nicola Tonellotto, and Iadh Ounis. 2021. Pseudo-relevance feedback for multiple representation dense retrieval. In *Proceedings of the ACM SIGIR International Conference on the Theory of Information Retrieval (ICTIR'21)*. 297–306.

[282] Yujing Wang, Yingyan Hou, Haonan Wang, Ziming Miao, Shibin Wu, Hao Sun, Qi Chen, Yuqing Xia, Chengmin Chi, Guoshuai Zhao, Zheng Liu, Xing Xie, Hao Allen Sun, Weiwei Deng, Qi Zhang, and Mao Yang. 2022. A neural corpus indexer for document retrieval. Retrieved from https://abs/2206.02743

[283] Yumeng Wang, Lijun Lyu, and Avishek Anand. 2022. BERT rankers are brittle: A study using adversarial document perturbations. In *Proceedings of the ACM SIGIR International Conference on the Theory of Information Retrieval (ICTIR'22)*. 115–120.

[284] Zhiguo Wang, Patrick Ng, Xiaofei Ma, Ramesh Nallapati, and Bing Xiang. 2019. Multi-passage BERT: A globally normalized bert model for open-domain question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP'19)*. 5878–5882.

[285] Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners. Retrieved from https://abs/2109.01652

[286] Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Scalable zero-shot entity linking with dense entity retrieval. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'20)*. 6397–6407.

[287] Xing Wu, Guangyuan Ma, Meng Lin, Zijia Lin, Zhongyuan Wang, and Songlin Hu. 2022. ConTextual mask auto-encoder for dense passage retrieval. Retrieved from https://arXiv:2208.07670

[288] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-end neural ad hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 55–64.

[289] 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 *Proceedings of the 9th International Conference on Learning Representations (ICLR'21)*.

[290] Canwen Xu, Daya Guo, Nan Duan, and Julian McAuley. 2022. LaPraDoR: Unsupervised pretrained dense retriever for zero-shot text retrieval. Retrieved from https://arXiv:2203.06169

[291] Shicheng Xu, Liang Pang, Huawei Shen, Xueqi Cheng, and Tat-seng Chua. 2023. Search-in-the-chain: Towards the accurate, credible and traceable content generation for complex knowledge-intensive tasks. Retrieved from https://arXiv:2304.14732

[292] Ikuya Yamada, Akari Asai, and Hannaneh Hajishirzi. 2021. Efficient passage retrieval with hashing for open-domain question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*. 979–986.

[293] Ming Yan, Chenliang Li, Chen Wu, Bin Bi, Wei Wang, Jiangnan Xia, and Luo Si. 2019. IDST at TREC 2019 deep learning track: deep cascade ranking with generation-based document expansion and pre-trained language modeling. In *Proceedings of the 28th Text Retrieval Conference (TREC'19) (NIST Special Publication, Vol. 1250)*.

[294] Yuanmeng Yan, Rumei Li, Sirui Wang, Fuzheng Zhang, Wei Wu, and Weiran Xu. 2021. ConSERT: A contrastive framework for self-supervised sentence representation transfer. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*. 5065–5075.

[295] Nan Yang, Furu Wei, Binxing Jiao, Daxing Jiang, and Linjun Yang. 2021. xMoCo: Cross momentum contrastive learning for open-domain question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*. 6120–6129.

[296] 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.

[297] Yinfei Yang, Ning Jin, Kuo Lin, Mandy Guo, and Daniel Cer. 2021. Neural retrieval for question answering with cross-attention supervised data augmentation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*. 263–268.

[298] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 2369–2380.

[299] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. Retrieved from https://arXiv:2210.03629

[300] Dani Yogatama, Cyprien de Masson d'Autume, and Lingpeng Kong. 2021. Adaptive semiparametric language models. *Trans. Assoc. Comput. Linguist.* 9 (2021), 362–373.

[301] HongChien Yu, Chenyan Xiong, and Jamie Callan. 2021. Improving query representations for dense retrieval with pseudo relevance feedback. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*. 3592–3596.

[302] Shi Yu, Zhenghao Liu, Chenyan Xiong, Tao Feng, and Zhiyuan Liu. 2021. Few-shot conversational dense retrieval. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'21)*. 829–838.

[303] Hansi Zeng, Hamed Zamani, and Vishwa Vinay. 2022. Curriculum learning for dense retrieval distillation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'22)*. 1979–1983.

[304] ChengXiang Zhai. 2008. *Statistical Language Models for Information Retrieval*. Morgan & Claypool Publishers.

[305] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Interpreting dense retrieval as mixture of topics. Retrieved from https://arXiv:2111.13957

[306] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Jointly optimizing query encoder and product quantization to improve retrieval performance. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*. 2487–2496.

[307] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Learning discrete representations via constrained clustering for effective and efficient dense retrieval. Retrieved from https://arXiv:2110.05789

[308] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng 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.

[309] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2020. RepBERT: Contextualized text embeddings for first-stage retrieval. Retrieved from https://abs/2006.15498

[310] Jingtao Zhan, Xiaohui Xie, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2022. Evaluating extrapolation performance of dense retrieval. Retrieved from https://abs/2204.11447

[311] Hang Zhang, Yeyun Gong, Yelong Shen, Jiancheng Lv, Nan Duan, and Weizhu Chen. 2021. Adversarial retriever-ranker for dense text retrieval. Retrieved from https://arXiv:2110.03611

[312] Han Zhang, Hongwei Shen, Yiming Qiu, Yunjiang Jiang, Songlin Wang, Sulong Xu, Yun Xiao, Bo Long, and Wen-Yun Yang. 2021. Joint learning of deep retrieval model and product quantization based embedding index. Retrieved from https://arXiv:2105.03933

[313] Jianjin Zhang, Zheng Liu, Weihao Han, Shitao Xiao, Rui Zheng, Yingxia Shao, Hao Sun, Hanqing Zhu, Premkumar Srinivasan, Denvy Deng, Qi Zhang, and Xing Xie. 2022. Uni-retriever: Towards learning the unified embedding based retriever in bing sponsored search. Retrieved from https://abs/2202.06212

[314] Kai Zhang, Chongyang Tao, Tao Shen, Can Xu, Xiubo Geng, Binxing Jiao, and Daxin Jiang. 2022. LED: Lexicon-enlightened dense retriever for large-scale retrieval. Retrieved from https://abs/2208.13661

[315] Michael J. Q. Zhang and Eunsol Choi. 2021. SituatedQA: Incorporating extra-linguistic contexts into QA. Retrieved from https://arXiv:2109.06157

[316] Shunyu Zhang, Yaobo Liang, Ming Gong, Daxin Jiang, and Nan Duan. 2022. Multi-view document representation learning for open-domain dense retrieval. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. 5990–6000.

[317] Xinyu Zhang, Xueguang Ma, Peng Shi, and Jimmy Lin. 2021. Mr. TyDi: A multi-lingual benchmark for dense retrieval. Retrieved from https://abs/2108.08787

[318] Yanzhao Zhang, Dingkun Long, Guangwei Xu, and Pengjun Xie. 2022. HLATR: Enhance multi-stage text retrieval with hybrid list aware transformer reranking. Retrieved from https://abs/2205.10569. https://doi.org/10.48550/arXiv.2205.10569

[319] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong et al. 2023. A survey of large language models. Retrieved from https://arXiv:2303.18223

[320] Wei Zhong, Jheng-Hong Yang, and Jimmy Lin. 2022. Evaluating token-level and passage-level dense retrieval models for math information retrieval. Retrieved from https://abs/2203.11163

[321] Chunting Zhou, Jiatao Gu, and Graham Neubig. 2020. Understanding knowledge distillation in non-autoregressive machine translation. In *Proceedings of the 8th International Conference on Learning Representations (ICLR'20)*.

[322] Jiawei Zhou, Xiaoguang Li, Lifeng Shang, Lan Luo, Ke Zhan, Enrui Hu, Xinyu Zhang, Hao Jiang, Zhao Cao, Fan Yu et al. 2022. Hyperlink-induced pre-training for passage retrieval in open-domain question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. 7135–7146.

[323] Kun Zhou, Yeyun Gong, Xiao Liu, Wayne Xin Zhao, Yelong Shen, Anlei Dong, Jingwen Lu, Rangan Majumder, Ji-Rong Wen, Nan Duan, and Weizhu Chen. 2022. SimANS: Simple ambiguous negatives sampling for dense text retrieval. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'22)*.

[324] Kun Zhou, Beichen Zhang, Xin Zhao, and Ji-Rong Wen. 2022. Debiased contrastive learning of unsupervised sentence representations. In *Proceedings of the Association for Computational Linguistics (ACL'22)*. 6120–6130.
[325] Yucheng Zhou, Tao Shen, Xiubo Geng, Chongyang Tao, Can Xu, Guodong Long, Binxing Jiao, and Daxin Jiang. 2022. Towards robust ranker for text retrieval. Retrieved from https://abs/2206.08063
[326] Yujia Zhou, Jing Yao, Zhicheng Dou, Ledell Wu, Peitian Zhang, and Ji-Rong Wen. 2022. Ultron: An ultimate retriever on corpus with a model-based indexer. Retrieved from https://abs/2208.09257
[327] Yujia Zhou, Jing Yao, Zhicheng Dou, Ledell Yu Wu, and Ji rong Wen. 2022. DynamicRetriever: A pre-training model-based IR system with neither sparse nor dense index. Retrieved from https://abs/2203.00537
[328] Mu Zhu. 2004. Recall, precision and average precision. *Dept. Stat. Actuar. Sci. (University of Waterloo, Waterloo)* 2, 30 (2004), 6.
[329] Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Zhicheng Dou, and Ji-Rong Wen. 2023. Large language models for information retrieval: A survey. Retrieved from https://arXiv:2308.07107
[330] Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. 2022. RankT5: Fine-tuning T5 for text ranking with ranking losses. Retrieved from https://abs/2210.10634. https://doi.org/10.48550/arXiv.2210.10634
[331] Shengyao Zhuang, Hang Li, and G. Zuccon. 2022. Implicit feedback for dense passage retrieval: A counterfactual approach. Retrieved from https://abs/2204.00718
[332] Shengyao Zhuang, Houxing Ren, Linjun Shou, Jian Pei, Ming Gong, Guido Zuccon, and Daxin Jiang. 2022. Bridging the gap between indexing and retrieval for differentiable search index with query generation. Retrieved from https://abs/2206.10128
[333] Shengyao Zhuang and Guido Zuccon. 2021. Dealing with typos for BERT-based passage retrieval and ranking. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'21)*. 2836–2842.
[334] Shengyao Zhuang and G. Zuccon. 2022. Asyncval: A toolkit for asynchronously validating dense retriever checkpoints during training. Retrieved from https://abs/2202.12510
[335] Shengyao Zhuang and Guido Zuccon. 2022. CharacterBERT and self-teaching for improving the robustness of dense retrievers on queries with typos. In *Proceedings of the 45rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1444–1454.
[336] Justin Zobel and Alistair Moffat. 2006. Inverted files for text search engines. *ACM Comput. Surv.* 38 (2006), 6.
[337] Justin Zobel, Alistair Moffat, and Kotagiri Ramamohanarao. 1998. Inverted files versus signature files for text indexing. *ACM Trans. Database Syst.* 23, 4 (1998), 453–490.

Received 19 January 2023; revised 12 October 2023; accepted 3 December 2023