Dense Text Retrieval based on Pretrained Language Models: A Survey

Wayne Xin Zhao, Jing Liu, Ruiyang Ren and Ji-Rong Wen

Abstract—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 classic 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 the text representation and model the relevance matching. The recent success of pretrained language models (PLMs) 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 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 referred to as *dense retrieval*, since it employs dense vectors (*a.k.a.*, embeddings) to represent the texts. Considering the rapid progress on dense retrieval, in this survey, we systematically review the recent advances on PLM-based dense retrieval. Different from previous surveys on dense retrieval, we take a new perspective to organize the related work by four major aspects, including architecture, training, indexing and integration, and summarize the mainstream techniques for each aspect. We thoroughly survey the literature, and include 300+ related reference papers on dense retrieval. To support our survey, we create a website for providing useful resources, and release a code repertory and toolkit for implementing dense retrieval models. This survey aims to provide a comprehensive, practical reference focused on the major progress for dense text retrieval.

Index Terms—Text retrieval, Dense retrieval; Pretrained language models

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 [1], [2], dialog system [3], [4], entity linking [5], [6] and Web search [7].

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 [8]. Among the early efforts, a significant achievement is the *vector space model* [9], [10] based on the "bag-of-words" assumption, representing both documents and queries as sparse term-based vectors[1]. To construct sparse vector representations, various term weighting methods have been designed and implemented, including the classic *tf-idf* method [11]–[13]. 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* [14], [15], which organizes the text content as term-oriented posting lists, for efficient text retrieval. In order to better understand the underlying retrieval mechanism, probabilistic relevance frameworks have been proposed for relevance modeling, exemplified by the classic BM25 model [16], [17]. Furthermore, statistical language modeling approaches [18] 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* [19], [20] 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 [21] 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 representation 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)* [22]–[24],

• *Wayne Xin Zhao, Ruiyang Ren and Ji-Rong Wen are with Renmin University of China; Jing Liu is with Baidu Inc., Beijing, China. E-mail: batmafly@gmail.com, reyon.ren@ruc.edu.cn, liujing46@baidu.com.*

1. In this survey, sparse vectors mainly refer to term-based vector representations.

which can be considered as initial explorations for dense retrieval techniques. Following the convention in [25], [26], we refer to these neural IR methods (before the use of pretrained language models) as *pre-BERT models*.

Recently, based on the powerful Transformer architecture [27], pretrained language models (PLM) [28] 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 recent four years, a large number of studies on PLM-based dense retrieval have been proposed [25], [26], [29], 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 [30]. 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[2]. Second, the availability of large-scale labeled retrieval datasets (*e.g.,* MS MARCO [32] and Natural Questions [33] datasets) makes it feasible to train (or fine-tune) capable PLMs for text retrieval. For example, TREC 2019 and 2020 Deep Learning Track [31], [34] release a training set of 0.367 million queries (derived from MS MARCO [32]) 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 [21], [22]). 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 [24]–[26], [29], [35]. For example, Guo et al. [24] review early contributions in neural IR, Cai et al. [29] summarize the major progress for the first-stage retrieval in three different paradigms, Lin et al. [25] present a review of text ranking methods mainly based on pretrained Transformers, Fan et al. [26] review the major progress for information retrieval based on pretraining methods, and Nicola Tonellotto [35] 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:

- Firstly, 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.
- Secondly, 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.
- Thirdly, we cover the recent advances on dense retrieval, and discuss several emerging research topics (*e.g.,* model based retrieval and representation enhanced pretraining) in detail.

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 based on PaddlePaddle[3] (including RocketQA [36], PAIR [37] and RocketQAv2 [38]), 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, and Section 3 presents the existing datasets and evaluation metrics for dense retrieval. As the core content, Section 4, 5, 6 and 7 review the mainstream architecture, training approach, index mechanism and retrieval pipeline for dense retrieval, where we will thoroughly discuss the recent progress in the four aspects. Then, we continue the discussion on the advanced topics (Section 8) and the applications (Section 9). Finally, we conclude this survey by summarizing the major findings and remaining issues in Section 10.

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 and Terminology

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, we refer to these tasks as *text retrieval* in a unified way.

2. An example query that is difficult to answer by lexical match can be given as: "*average salary for dental hygienist in nebraska*". See more query examples in the TREC deep learning track report [31].

3. https://github.com/PaddlePaddle/RocketQA

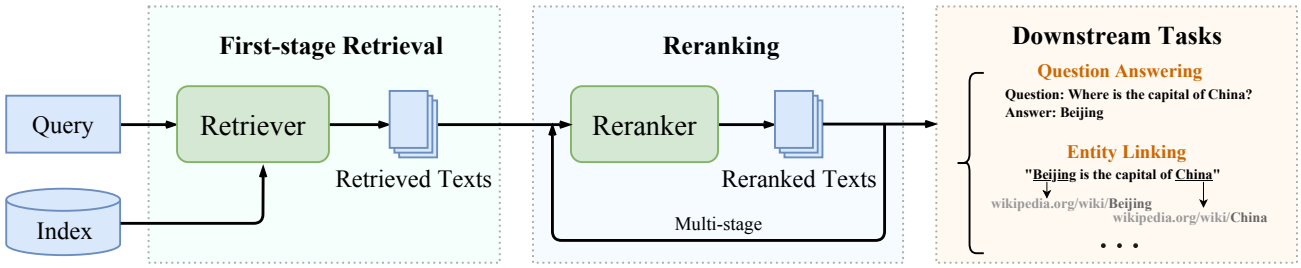


Fig. 1. The illustration for the overall pipeline of an information retrieval system.

Typically, a complete information retrieval system consists of multiple procedures or stages arranged in a processing pipeline [25], [26][4]. 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* [25], [29]. Correspondingly, the subsequent stages mainly focus on *reranking* (or *ranking*) the candidate texts, called *reranking stages*, which are supported by *rerankers* [25]. 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* [39] may be further integrated to analyze the returned texts by the retriever (or reranker) and locate the answers for the query.

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 of a retrieval model. As the technical approach, either sparse retrieval models (relying on 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 [25], [26], [29], [40], denoted by

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

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[5]. For dense retrieval, $\phi(\cdot)$ and $\psi(\cdot)$ are developed based on neural network encoders, and similarity measurement functions (*e.g.,* inner product) can be used to implement $f_{\text{sim}}(\cdot)$. At the pre-BERT time[6], the encoders are usually implemented by multi-layered perceptron networks (or other neural networks), while in this survey we focus on the text encoder based on pretrained language models (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 5.2). Note that in some cases, we can also obtain graded relevance judgements [42], [43], 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.

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 4). 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 5). 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 6). 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 7). 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.

4. For ease of discussion, we omit other parts such as data collection and preprocessing.

5. This formulation can be also considered as a generalized representational approach [40] to information retrieval, for both dense and sparse retrieval models.

6. Following the naming conventions in [41], we refer to the neural information retrieval methods as *pre-BERT approaches*, which are before the use of Transformer and BERT family.

3 Datasets, Evaluation and Resources

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

3.1 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 on dense retrieval. We categorize the available retrieval datasets into three major categories according to their original tasks, namely information retrieval, question answering and other tasks. The statistics of the available datasets are shown in Table 1.

As we can see from Table 1, Wikipedia and Web are two major resources for creating these datasets. Among these datasets, *MS MARCO* dataset [32] contains a large amount of queries with annotated relevant passages in Web documents, and *Natural Questions (NQ)* dataset [33] contains Google search queries with annotations (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 [70] 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 in order to enrich the evaluation characteristics on some specific aspect, *e.g.,* the multilingual version mMARCO [44] and the MS MARCO Chameleons dataset [71] (consisting of obstinate queries that are difficult to answer by neural retrievers despite having the similar query length and distribution of relevance judgements as queries that are easier to answer). Besides, Sciavolino et al. [52] 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 [45], financial dataset [60], biomedical dataset [61], climate-specific dataset [67] and scientific dataset [68].

Besides the presented datasets in Table 1, 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* [72] and *KILT* [73].

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 [74], 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. Incomplete relevance annotations lead to potential training issues such as false negative, which would harm 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 [54] releases a large-scale Chinese dataset consisting of 90K queries from Baidu search and over 8M passages for passage retrieval. In order 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, it also provides human-translated queries (in English) for cross-lingual retrieval.

3.2 Evaluation Metrics

To evaluate the retrieval capacity of an information retrieval system, a number of factors need to be considered [75], [75], [76]: effectiveness, efficiency, diversity, novelty and so on. This survey mainly focuses on the effectiveness for the retrieval system[7]. We next introduce the commonly 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 [77] calculates a truncated Recall value at the k-th position of a retrieved list:

$$\text{Recall@}k = \frac{1}{|\mathcal{Q}|}\sum_{q=1}^{|\mathcal{Q}|}\frac{\#\text{retr}_{q,k}}{\#\text{rel}_q}, \tag{2}$$

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 $\mathcal{Q}$.

In dense retrieval, there is a commonly used metric, *Top-k Accuracy*, for computing the proportion of queries for which the top-k retrieved texts contain the answers [1], defined as:

$$\text{Accuracy@}k = \frac{1}{|\mathcal{Q}|}\sum_{q=1}^{|\mathcal{Q}|}\mathbb{I}(\#\text{retr}_{q,k} > 0), \tag{3}$$

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 [1], [36], a retrieved list is considered to *accurately* solve a query when it contains the answer, not necessarily retrieving all the relevant texts [8].

7. 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.

8. Note that Accuracy@k is also known as Recall@k in some dense retrieval studies [36], [78], which is somehow different from the definition used in traditional retrieval benchmarks.

TABLE 1
Detailed statistics of available retrieval datasets. Here, "q" is the abbreviation of queries, and "instance" denotes a candidate text in the collection.

Categorization	Domain	Dataset	#q in train	#label in train	#q in dev	#q in test	#instance
Information retrieval	Web	MS MARCO [32]	502,939	532,761	6,980	6,837	8,841,823
	Web	mMARCO [44]	808,731	–	101,093	–	8,841,823
	News	TREC-NEWS [43]	–	–	–	57	594,977
	Biomedical	TREC-COVID [45]	–	–	–	50	171,332
	Biomedical	NFCorpus [46]	5,922	110,575	324	323	3,633
	Twitter	Signal-1M [47]	–	–	–	97	2,866,316
	Argument	Touché-2020 [48]	–	–	–	249	528,155
	Argument	ArguAna [49]	–	–	–	1,406	8,674
	Wikipedia	DBPedia [50]	–	–	67	400	4,635,922
	Web	ORCAS [51]	10.4M	18.8M	–	–	3,213,835
	Wikipedia	EntityQuestions [52]	176,560	186,367	22,068	22,075	–
	Web	MS MARCO v2 [53]	277,144	284,212	8,184	–	138,364,198
	Web	$\text{DuReader}_{\text{retrieval}}$ [54]	97,343	86,395	2,000	8,948	8,096,668
Question answering	Wikipedia	Natural Questions [33]	152,148	152,148	6,515	3,610	2,681,468
	Wikipedia	SQuAD [55]	78,713	78,713	8,886	10,570	23,215
	Wikipedia	TriviaQA [56]	78,785	78,785	8,837	11,313	740K
	Wikipedia	HotpotQA [57]	85,000	170,000	5,447	7,405	5,233,329
	Web	WebQuestions [58]	3,417	3,417	361	2,032	–
	Web	CuratedTREC [59]	1,353	1,353	133	694	–
	Finance	FiQA-2018 [60]	5,500	14,166	500	648	57,638
	Biomedical	BioASQ [61]	3,743	35,285	–	497	15,559,157
	StackEx.	CQADupStack [62]	–	–	–	13,145	457,199
	Quora	Quora [63]	–	–	5,000	10,000	522,931
	News	ArchivalQA [64]	853,644	853,644	106,706	106,706	483,604
	Web	CCQA [65]	55M	130M	–	–	–
Other tasks	Wikipedia	FEVER [66]	–	140,085	6,666	6,666	5,416,568
	Wikipedia	Climate-FEVER [67]	–	–	–	1,535	5,416,593
	Scitific	SciFact [68]	809	920	–	300	5,183
	Scitific	SciDocs [69]	–	–	–	1,000	25,657

Besides, Precision@k [77] calculates the average proportion of relevant texts among the top k positions over the query set $\mathcal{Q}$:

$$\text{Precision@}k = \frac{1}{|\mathcal{Q}|}\sum_{q=1}^{|\mathcal{Q}|}\frac{\#\text{retr}_{q,k}}{k}. \tag{4}$$

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

$$\text{AP}_q = \frac{1}{\#\text{rel}_q}\sum_{k=1}^{L}\text{Precision@}k \times \mathbb{I}(q,k), \tag{5}$$

where Precision@k is the per-query version of the precision value at the k-th position, L is the length of a retrieved list and $\mathbb{I}(q,k)$ is an indicator function returning 1 only when the k-th position corresponds to a relevant text for query q. Furthermore, Mean Average Precision (MAP) [77] calculates the mean of Precision scores over a set of queries Q:

$$\text{MAP} = \frac{1}{|\mathcal{Q}|}\sum_{q=1}^{|\mathcal{Q}|}\text{AP}_q. \tag{6}$$

Not only counting the occurrence of positive texts, DCG (Normalized Discounted Cumulative Gain) [42] further incorporates the position of a relevant text into consideration, and it prefers a ranking that places a relevant text at a higher position:

$$\text{DCG}_q@k = \sum_{i=1}^{k}\frac{2^{g_i}-1}{\log_2(i+1)}, \tag{7}$$

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:

$$\text{NDCG@}k = \frac{1}{|\mathcal{Q}|}\sum_{q=1}^{|\mathcal{Q}|}\frac{\text{DCG}_q@k}{\text{IDCG}_q@k}, \tag{8}$$

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

Furthermore, MRR [2] averages the reciprocal of the rank of the first retrieved positive text over a set of queries $\mathcal{Q}$:

$$\text{MRR} = \frac{1}{|\mathcal{Q}|}\sum_{q=1}^{|\mathcal{Q}|}\frac{1}{\text{rank}_q}, \tag{9}$$

where 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 [75], [76], [79], [80].

3.3 Code Library Resource

Recently, several open-sourced dense retrieval libraries have been released for research purpose. As a representative library, Tevatron [81] 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 [82] 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 [83] and FAISS [84]. It also provides the evaluation scripts for the standard IR test collections. It can be accessed at the link: http://pyserini.io/.

In order to enhance the validation of dense retriever checkpoints, Asyncval [85] 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 accessed at the link: https://github.com/ielab/asyncval.

OpenMatch [86] has been originally proposed for pre-BERT neural IR models (v1), and extended (v2) to support dense retrieval models on commonly used benchmarks such as MS MARCO and NQ. It can be accessed at the link: https://github.com/thunlp/OpenMatch.

MatchZoo [87] 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.

PyTerrier [88] is a retrieval framework that supports the declarative use of Python operators for building retrieval pipelines and evaluating retrieval models, covering representative learning-to-rank, neural reranking and dense retrieval models. It can be accessed at the link: https://github.com/terrier-org/pyterrier.

SentenceTransformers [89] is another library that provides an easy way to compute dense embeddings for sentences and paragraphs based on Transformer networks. Specifically, it integrates the implementation of Sentence-BERT [89] and Transformer-based Sequential Denoising Auto-Encoder (TSDAE) [90]. It can be accessed at the link: https://www.sbert.net/.

As the supporting resource, we also release an open-sourced implementation of dense retrievers based on our previous work RocketQA [36], at the link: https://github.com/PaddlePaddle/RocketQA, including RocketQA [36], RocketQA$_{PAIR}$ [37] and RocketQAv2 [38]. This software also provides an easy-to-use toolkit with pre-built models (including both English and Chinese models) for direct use after installation. We also aggregate other open-sourced codes for related dense retrieval papers in our survey website, which can be found in the link: https://github.com/RUCAIBox/DenseRetrieval.

4 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.

4.1 Background

4.1.1 Transformer and Pretrained Language Models

Transformer [27] 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 [156], [157]), 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 support of GPU or TPU. As a remarkable contribution, Transformer makes it more flexible and easier to train very large neural networks. There are a large number of variants that improve the basic Transformer architecture, and the readers can refer to the related surveys [158], [159] for a comprehensive discussion.

Based on Transformer and its variants, pretrained language models (PLM) are proposed [28], [160], [161] in the field of natural langauge 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 various downstream tasks. Such a learning approach is called *pretraining then fine-tuning paradigm* [28], [162], which is one of the most striking achievements on language intelligence in recent years. As one of the most representative PLMs, BERT [28] 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. The success of BERT inspires a series of follow-up studies, including the improved pretraining approach [160], the refined bidirectional representations [163], and model compression [164]. Furthermore, several studies attempt to train extremely large PLMs in order to explore the performance limit as a universal learner [161], [165], achieving very impressive results on downstream tasks.

Beyond language modeling, PLMs have so far become a general technical approach to model, represent and manipulate large-scale unlabeled data, which are also called *foundation models* [166]. A detailed introduction of recent progress on foundation models is beyond the scope of this survey, which is omitted in this survey.

4.1.2 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

TABLE 2
A detailed list of different dense retrieval methods in the literature with detailed configurations (abbreviations are from Table 3).

Years	Methods	PLMs	Arch.	Training				Other Tricks
				Loss	Negative	Data Aug.	Pretrain	
2020	Poly-encoders [91]	$BERT_{base}$	MR	CE			TAP	Context-candidate attention
2019	ICT [92]	$BERT_{base}$	SR	IOC			TAP	Joint training
2020	ICT+BFS+WLP [93]	$BERT_{base}$	SR	IOC			TAP	
2020	REALM [94]	$BERT_{base}$	SR	IOC			RAP	Joint training + Async. index
2020	DPR [1]	$BERT_{base}$	SR	NLL	IB+SHN	ALD		
2020	ColBERT [95]	$BERT_{base}$	MR	CE				Multi-core pre-processing
2020	ME-BERT [96]	$BERT_{large}$	MR	CE	IB+DyHN			Sparse-dense hybrid
2020	RepBERT [97]	$BERT_{base}$	SR	IOC	IB			
2020	ANCE [98]	$RoBERTa_{base}$	SR	NLL	DyHN			Async. index
2020	EZRQG [99]	$BERT_{base}$	SR	IOC				Query generation
2020	AugmentedBERT [100]	$BERT_{base}$	SR	IOC+KL	SHN	KD		Kernel density estimation
2020	In-batch KD [101]	$BERT_{base}$	SR	MSE	IB	KD		Cross-architecture KD
2020	RocketQA [36]	$ERNIE_{base}$	SR	NLL	CB+DeHN	KD		Iter. training
2020	AugmentSBERT [102]	$BERT_{base}$	SR	NLL	SHN+DeHN	KD		Kernel density estimation
2020	TCT-ColBERT [103]	$BERT_{base}$	SR	KL	IB+SHN	KD		Sparse-dense hybrid
2020	Multi-stage [104]	$BERT_{base}$	SR	NLL	IB+SHN	KD	TAP	Embedding fusion
2020	DKRR [105]	$BERT_{base}$	SR	KL	SHN	KD		Iter. training
2020	DensePhrases [106]	$SpanBERT_{base}$	PR	IOC+KL	IB	KD		Pre-batch negatives
2020	UnifiedQA [107]	$BERT_{base}$	SR	NLL	IB	ALD		
2021	Individual Top-k [108]	$BERT_{large}$	SR	IOC+NLL	IB+SHN		TAP	Joint training
2021	Grad. Cache [109]	$BERT_{base}$	SR	NLL	CB			Gradient caching
2021	TAS-Balanced [110]	$BERT_{base}$	SR	MSE	IB+SHN	KD		Balanced topic aware sampling
2021	ADORE+STAR [111]	$RoBERTa_{base}$	SR	NLL	SHN+DyHN			Async. index
2021	Condenser [112]	Condenser	SR	CE	IB		REP	
2021	DRPQ [113]	$BERT_{base}$	MR	NLL	IB+DyHN			Multiple pseudo query embeddings
2021	RANCE [114]	$RoBERTa_{base}$	SR	NLL	DeHN			
2021	DANCE [115]	$RoBERTa_{base}$	SR	IOC	SHN			Contrastive dual learning
2021	DPR-PAQ [116]	$RoBERTa_{large}$	SR	NLL	IB+SHN	ALD	GAP	
2021	JPQ [117]	$BERT_{base}$	SR	IOC	DyHN			Quantization
2021	coCodenser [78]	Condenser	SR	NLL	SHN		REP	Corpus-aware pretraining
2021	PAIR [37]	$ERNIE_{base}$	SR	IOC+NLL	IB+SHN	KD	REP	Passage-centric constrain
2021	ST5 [118]	$T5_{11B}$	SR	NLL	IB			
2021	ANCE-PRF [119]	$RoBERTa_{base}$	SR	NLL	DyHN			Pseudo relevance feedback
2021	Trans-Encoder [120]	$RoBERTa_{base}$	SR	MSE	DyHN	KD		Iter. training + Mutual distillation
2021	AR2-G [121]	$ERNIE_{base}$	SR	IOC	DyHN			Adversarial training
2021	RepCONC [122]	$BERT_{base}$	SR	IOC	DyHN			Quantization
2021	RocketQAv2 [38]	$ERNIE_{base}$	SR	KL	SHN+DeHN	KD		Joint training + Mutual distillation
2021	SPAR [123]	$BERT_{base}$	SR	NLL	SHN	KD		
2021	SEED [124]	$BERT_{base}$	SR	NLL	DyHN		TAP	Autoencoder pretraining
2021	ColBERTv2 [125]	$BERT_{base}$	SR	NLL	DeHN	KD		Residual compression + Quantization
2021	GPL [126]	$DistilBERT_{base}$	SR	MSE	IB+SHN	KD	TAP	Query generation
2021	Spider [127]	$BERT_{base}$	SR	NLL	IB+SHN		TAP	Sparse-dense hybrid
2021	DrBoost [128]	$BERT_{base}$	SR	NLL	SHN			Boosting
2021	GTR [129]	$T5_{11B}$	SR	NLL	IB+DeHN		TAP	
2021	Contriever [130]	$BERT_{base}$	SR	NLL	IB		REP	Contrastive learning
2022	Uni-Retriever [131]	$BERT_{base}$	SR	IOC+NLL	CB + SHN			
2022	LaPraDoR [132]	$DistilBERT_{base}$	SR	IOC+NLL	IB		REP	Sparse-dense hybrid
2022	HLP [133]	$BERT_{base}$	SR	NLL	IB		TAP	
2022	DAR [134]	$RoBERTa_{large}$	SR	NLL	IB+SHN			Interpolation and perturbation
2022	MVR [135]	$BERT_{base}$	MR	IOC+NLL	SHN			Multi-view representation
2022	ColBERTer [136]	$DistilBERT_{base}$	MR	IOC+MSE	SHN			
2022	CharacterBERT [137]	CharacterBERT	SR	NLL+KL	IB+SHN			Self-teaching
2022	GNN-encoder [138]	$ERNIE_{base}$	SR	IOC+NLL	IB+DeHN	KD		Query-passage interaction
2022	COSTA [139]	$BERT_{base}$	SR	NLL	SHN		TAP	Contrastive span prediction
2022	CL-DRD [140]	$DistilBERT_{base}$	SR	KL	KD			Curriculum Learning
2022	ERNIE-Search [141]	$ERNIE_{large}$	SR	NLL+KL	IB	KD		Joint training + Mutual distillation
2022	RetroMAE [142]	$BERT_{base}$	SR	IOC+NLL	DyHN	KD	REP	MAE pretraining
2022	DCSR [143]	$BERT_{base}$	MR	NLL	IB+SHN			
2022	ART [144]	$BERT_{large}$	SR	KL				Query reconstruction
2022	SimLM [145]	$BERT_{base}$	SR	NLL	IB+DeHN	KD	REP	Iter. training
2022	RoDR [146]	$BERT_{base}$	SR	NLL+KL	IB+SHN			Passage-centric constraint
2022	Aggretriever [147]	Condenser	SR	NLL	IB+SHN			Text aggregation
2022	CoT-MAE [148]	$BERT_{base}$	SR	NLL	SHN		GAP	
2022	CPDAE [149]	$BERT_{base}$	SR	IOC+NLL	SHN		TAP	Autoencoder pretraining
2022	DPTDR [150]	$RoBERTa_{large}$	SR	NLL	IB+DeHN	KD	TAP	Deep prompt tuning
2022	LED [151]	Condenser	SR	IOC+NLL	IB+SHN	KD		Rank-consistent regularization
2022	LexMAE [152]	$BERT_{base}$	SR	IOC+NLL+KL	SHN	KD	REP	MAE pretraining
2022	Promptagator [153]	FLAN	SR	NLL	IB+SHN			Prompt-base query generation
2022	PROD [154]	$ERNIE_{base}$	SR	NLL+KL	IB+SHN	KD		Curriculum learning
2022	TASER [155]	Condenser	SR	NLL	IB+SHN			Mixture-of-experts

TABLE 3
Abbreviations for different techniques or strategies.

Type				Abbr.
Architecture	Single-representation (Sec. 4.2.2)			SR
	Multi-representation (Sec. 4.2.2)			MR
	Phrase-level representation (Sec. 4.2.2)			PR
Training	Loss	Negative log-likelihood loss (Sec. 5.1.1)		NLL
		Cross-entropy loss (Sec. 5.1.1)		CE
		Triplet ranking loss (Sec. 5.1.1)		TR
		MSE loss (Sec. 5.3.2)		MSE
		KL-divergence loss (Sec. 5.3.2)		KL
		Incorporating optimization constraints (Sec. 5.3.2)		IOC
	Negative selection	In-batch negatives (Sec. 5.2.1)		IB
		Cross-batch negatives (Sec. 5.2.2)		CB
		Hard negatives	Static hard negatives (Sec. 5.2.3)	SHN
			Dynamic hard negatives (Sec. 5.2.3)	DyHN
			Denoised hard negatives (Sec. 5.2.3)	DeHN
	Data augmentation	Auxiliary labeled datasets (Sec. 5.3.1)		ALD
		Knowledge distillation (Sec. 5.3.2)		KD
	Pretraining	Task adaptive pretraining (Sec. 5.4.1)		TAP
		Generation-augmented pretraining (Sec. 5.4.2)		GAP
		Retrieval-augmented pretraining (Sec. 5.4.3)		RAP
		Representation enhanced pretraining (Sec. 5.4.4)		REP

a representative study, DeepCT [167] 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 [168] 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, 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 [169] 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[9]. To better understand the above models, Lin et al. [170] 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 [170] is also proposed by reducing the weight vector in COIL to one dimension, and the experiments in [170] 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 [171] 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 [172] and SPLADEv2 [173] 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, so as 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 [169]). For a more thorough discussion of PLM-enhanced sparse retrieval models, the readers can refer to [25], [26], [29].

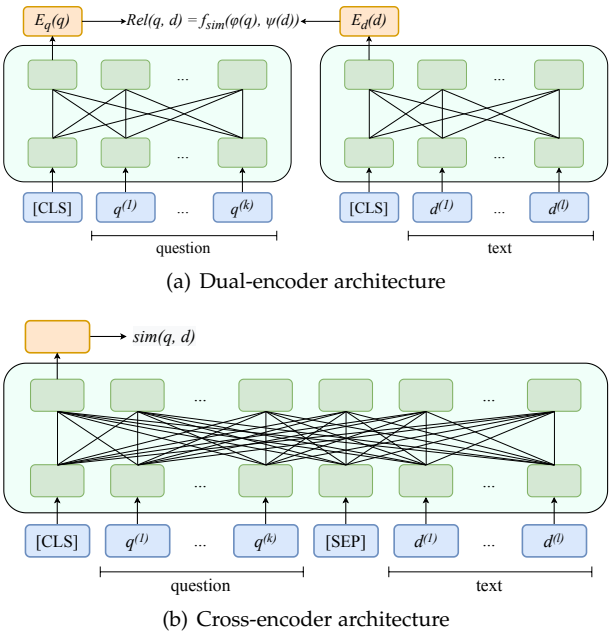
4.2 Neural Architecture for Dense Retrieval

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 ways, there are two mainstream architectures for dense retrieval, namely the cross-encoder and bi-encoder.

4.2.1 The 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]". In order 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

9. In order to alleviate the term mismatching problem, COIL also incorporates semantic matching based on "[CLS]" embeddings.



(a) Dual-encoder architecture

(b) Cross-encoder architecture

Fig. 2. Visual illustration of dual-encoder and cross-encoder architectures.

used way is to take the "[CLS]" representation as the semantic matching representation [174]. Other variants can be also used, *e.g.*, averaging token embeddings [89]. Such an architecture is similar to *interaction-based architecture* in the pre-BERT studies [22], [175], since it allows the tokens to interact across queries and texts.

4.2.2 The 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 [21]) 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 [1], [36], [98] learn the query and text representations with encoder-only PLMs (*e.g.*, BERT [28], RoBERTa [28], and ERNIE [176]). More recently, text-to-text Transformer (T5) [165], which is an encoder-decoder based PLM, has been explored to learn text representations for dense retrieval [118], [129]. It has been shown that a T5-based sentence embedding model outperforms Sentence-BERT [177] on SentEval [178] and SentGLUE [165] 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 [91] 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 [96] 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 [95] 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 4.2.1), 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* [95]. As the extension of ColBERT, ColBERTer [136] designs an approach by combining single-representation (the "[CLS]" embedding) and multi-representation (per-token embeddings) mechanisms for matching, achieving performance improvement over ColBERT. More recently, MVR [135] proposes to insert multiple "[VIEW]" tokens (similar to "[CLS]") at the beginning of a text, and aims to learn the text representations in multiple different views. Further, they design a special local loss to distinguish the best matched view from the rest views of a text for a query. MADRM [179] proposes to learn multiple aspect embeddings for both query and text, and employ explicit aspect annotations to supervise the aspect learning. Instead of using standard dense retrieval benchmarks, this work conducts the evaluation experiments based on the e-commerce datasets.

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 [136] 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 [136] 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 [106]. Recently, several studies propose to learn phrase-level representations for directly retrieving phrases as the answers to queries [106], [180]–[182]. The basic idea is to preprocess all the documents in a query-agnostic way and generate phrase-level represenations (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) [182] presents the first approach that leverages dense phrase index for question answering. DenSPI [180] further combines dense and sparse vectors of phrases to capture both semantic and lexical information, so as 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. [106] 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 [181] is that a dense phrase retrieval system can achieve a better passage retrieval accuracy than DPR on NQ and TriviaQA, without any retraining. 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 the above major representation ways, there are also several improvement variants in different aspects.

• *Composite architectures*. The above bi-encoders and variants can be applied for either document retrieval or passage retrieval, depending on specific tasks. We can also devise composite architectures by integrating text encoders in different semantic granularities. In a recent work [183], a hierarchical retrieval approach is proposed by combining a document retriever and a passage retriever, considering both document-level and passage-specific semantics. Besides, a boosting approach has been proposed in [128] that ensembles multiple "weak" dense retrievers for achieving a good retrieval performance. Further, Li et al. [184] propose to combine multiple dense retrievers trained with different tasks in an uncertainty-weighted way.

• *Lightweight architectures*. Besides retrieval performance, several approaches aim to learn lightweight representations, including the query embedding pruning methods [185] according to the estimated term importance, lightweight late-interaction variant [125] (*i.e., ColBERTv2*) that utilizes cross-encoder distillation with hard negatives and residual representation compression, and product quantization approaches [117] that uses discrete quantization representations (detailed in Section 6.2.3).

• *Incorporating relevance feedback*. As a classic strategy, pseudo relevance feedback has been re-visited in the setting of dense retrieval to enhance the query representation [119], [186]–[188]. The basic idea is to utilize the initially retrieved documents to derive the enhanced query embedding. Furthermore, Zhuang et al. [189] explore click-through data as feedback signal by using simulation based on click models, and then propose a counterfactual approach to dealing with the bias issue in click data. Wang et al. [190] employ relevance feedback to enhance the multi-representation dense retrieval models, showing a large performance improvement over the base model ColBERT.

• *Parameter-efficient tuning*. Since it is costly to fine-tune large-scale PLMs, parameter-efficient tuning [162] has been proposed for adapting PLM to the retrieval task [150], [191]. Generally, parameter-efficient tuning tends to underperform fine-tuning methods when applied directly to dense retrieval. Considering this issue, DPTDR introduces retrieval-oriented pretraining and negative mining [150] for enhancing the parameter-efficient tuning, which achieves a comparable performance as fine-tuned retrievers. Besides, the experiments in [191] have shown that parameter-efficient tuning (*e.g.,* P-tuning v2 [192]) can improve the zero-shot retrieval performance of DPR and ColBERT.

4.2.3 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 [22], 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. In order to have a deeper understanding, we next compare cross-encoder and bi-encoder from the perspective of dense text retrieval.

Firstly, 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 [36], [100], [193]. 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 [91], [95]. According to the reported results on MS MARCO [38], [125], 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 [180]. For efficiency, it can be accelerated via approximate nearest neighbor search for large-scale vector recall [84], which is particularly important for practical use of dense retrieval approaches.

Considering the pros and cons of the two architectures, in practice, 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 7.1.2). Besides, cross-encoder is often utilized to improve the bi-encoder, *e.g.,* knowledge distillation [38], [102] (Section 5.3.2) and pseudo labeled data generation [36], [37] (Section 5.4.2).

4.3 Sparse Retrieval *v.s.* Dense 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.

4.3.1 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 Do dense retrieval models always outperform sparse retrieval models? What are their respective strengths and weaknesses?

On the one hand, in the early literature of dense retrieval [1], [36], [98], 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, Natural Questions and TriviaQA) 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. Thus, sparse retrievers are unable to well resolve these queries due to the incapability in handling the term mismatch issue. By contrast, dense retrievers are more effective to answer complex queries by measuring the latent semantic similarity. Take an example query that cannot be resolved by a sparse retriever from the DPR paper [1]: 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*" [1].

On the other hand, 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 [72]. 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 [72]. Hence, a number of studies are developed to improve the zero-shot retrieval capabilities of dense retrievers (see Section 8.1 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 [52], [194]. For example, the DPR paper [1] 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 4.3.2 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?

Dense retrieval employs latent semantic representations for relevance matching, showing different retrieval performance compared with sparse retrieval models: they perform better on complicated 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 they perform exact match). For this purpose, Luan et al. [96] 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. Besides, another related study shows that corpus scale has a larger effect on dense retrieval [195]: "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 [196]–[198]. Specifically, IR axioms are formalized as a set of relevance constraints that reasonable IR models are desired to satisfy or at least partially satisfy [197], [199], *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, it has been found that existing IR axioms may only partially explain [200] or even not be suitable to analyze [201] the behavior of PLM-based IR models. Some further analysis [202] 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 [203] also shows that some dense retrieval model (*e.g.,* ColBERT [95]) is aware of term importance in text representations, and can also implicitly mimic exact matching for important terms.

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

4.3.2 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 [1], [204], while PLMs can be also used to learn term weights to improve sparse retrieval [167], [168] (as discussed in Section 4.1.2). In particular, it also shows that sparse retrieval models can be utilized to enhance the zero-shot retrieval capacity of dense retrieval models (see more discussions in Section 8.1).

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 [1], [96]. Further, a classifier is employed to select among sparse, dense, or hybrid retrieval strategies specially for each individual query [205], 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. [147], [206] propose to learn low-dimensional dense lexical representations (DLRs) 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 [147], [206] 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. [123] 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 and RocketQA) 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, so as to improve the robustness [123].

Besides, LED [151] proposes to employ PLM-based lexical-aware models (*i.e.,* SPLADE [172]) to enhance the lexical matching capacity of dense retrievers. It introduces lexicon-augmented contrastive learning and rank-consistent regularization for developing the lexicon-enhanced dense retrieval models. The experiment results show that the proposed approach outperforms a number of competitive baselines.

5 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.

5.1 Formulation and Training Issues

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

5.1.1 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)[10] 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:

$$\begin{aligned} & \mathcal{L}(\langle q_i, d_i^+ \rangle) \\ = \; & -\log \frac{e^{f(\phi(q_i), \psi(d_i^+))}}{e^{f(\phi(q_i), \psi(d_i^+))} + \sum_{d' \in \mathcal{D}^-} e^{f(\phi(q_i), \psi(d'))}}, \end{aligned} \tag{10}$$

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. Since the normalization term is time-consuming to compute, a negative sampling trick is usually adopted to reduce the computational cost [1], [36]:

$$\begin{aligned} & \mathcal{L}(\langle q_i, d_i^+ \rangle) \\ = \; & -\log \frac{e^{f(\phi(q_i), \psi(d_i^+))}}{e^{f(\phi(q_i), \psi(d_i^+))} + \sum_{d' \in \mathcal{N}_{q_i}} e^{f(\phi(q_i), \psi(d'))}}, \end{aligned} \tag{11}$$

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 [207] 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 [1], [208] directly optimizes

10. As we introduced in Section 2.2, only binary relevance judgements are considered here, which are the most common form of user feedback data in real-world search systems.

the difference between a positive text and a negative text given a query, which is defined as:

$$\begin{aligned} & \mathcal{L}(\langle q_i, d_i^+, d_i^- \rangle) \\ = \ & \max\Big(0, 1 - (f(\phi(q_i), \psi(d_i^+)) - f(\phi(q_i), \psi(d_i^-)))\Big). \end{aligned} \tag{12}$$

It has been reported that negative log-likelihood loss is better than the triplet ranking loss [1] 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 [193] (detailed in Section 7). 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:

$$\Pr(\text{rel} = 1 | q_i, d_i) = \sigma\big(g(\phi(q_i) \odot \psi(d_i))\big), \tag{13}$$

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:

$$\begin{aligned} & \mathcal{L}(\langle q_i, d_i \rangle) \\ = \ & -y_{q_i,d_i} \cdot \Pr_{q_i,d_i} - (1 - y_{q_i,d_i}) \cdot (1 - \Pr_{q_i,d_i}), \end{aligned} \tag{14}$$

where d_i can be either positive or negative, and $\Pr_{q_i,d_i} = \Pr(\text{rel} = 1 | q_i, d_i)$ which is defined in Equation (13).

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. [72] 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. [1] 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.

5.1.2 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 (11)) 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:

$$\begin{aligned} & \mathcal{L}_T(\langle q_i, d_i^+ \rangle) \\ = \ & -\log \frac{e^{f(\psi(d_i^+), \phi(q_i))}}{e^{f(\psi(d_i^+), \phi(q_i))} + \sum_{q^- \in \mathcal{Q}^-} e^{f(\psi(d_i^+), \phi(q^-))}}, \end{aligned} \tag{15}$$

where a set of sampled negative queries denoted by $\mathcal{Q}^-$ 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 Equation (11) and Equation (15). Similar symmetric optimization tricks have been used in a number of follow-up studies [115], [132], [209], [210], showing an improved retrieval performance.

Text-text similarity constraints. Besides directly optimizing query-text similarities, Ren et al. [37] 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:

$$\begin{aligned} & \mathcal{L}_{TT}(\langle q_i, d_i^+ \rangle) \\ = \ & -\log \frac{e^{f(\psi(d_i^+), \phi(q_i))}}{e^{f(\psi(d_i^+), \phi(q_i))} + \underline{\sum_{d' \in \mathcal{N}_{q_i}} e^{f(\psi(d_i^+), \psi(d'))}}}. \end{aligned} \tag{16}$$

The major difference between Equation (11) and Equation (16) 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.

5.1.3 Major Training Issues

In this part, we summarize the major training issues of dense retrieval.

• Issue 1: *Large-scale candidate space*. In retrieval tasks, there is typically a large number of candidates in a text collection, while only a few texts from the collection 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 space and time limits, we can sample only a small number of negative samples for computing the loss function (Equation (11)), resulting in a shift in the candidate space between training (a small population of texts) and testing (the entire collection). It has been found the sampled negatives have a significant effect on the retrieval performance [1], [36], [98].

• 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, which has become a major challenge for training dense retrievers.

• Issue 3: *Pretraining discrepancy*. PLMs are typically trained using pre-designed self-supervised loss functions, such as masked word prediction and next sentence prediction [28]. These pretraining tasks are not specially optimized for the retrieval tasks [93], [127], [211], 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. 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 5.2 (negative selection), Section 5.3 (data augmentation) and Section 5.4 (pretraining).

5.2 Negative Selection

To optimize the dense retriever (Equation (11)), 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.

5.2.1 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 [212] for the response selection task of *Smart Reply*, while it has been explicitly utilized for dense retrieval by DPR [1]. In-batch negatives are shown to be effective to improve the learning of bi-encoder by increasing the number of negatives [1], [36].

5.2.2 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. In order to further optimize the training process with more negatives, another improvement strategy called *cross-batch negatives* [36], [109] 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 [109], where one can accumulate multiple mini-batches for increasing the number of negatives (while taking a larger computational cost).

5.2.3 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 [1], [36], [98]. 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* [98], [111]. 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 [36]. 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 [213] 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 [1]. In [204], 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 [104] further design three fusion strategies to combine different kinds of negatives, namely mixing fusion, embedding fusion and rank fusion. Besides, Hofstatter et al. [110] 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 [98] 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 [98] 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. [111] 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 [98], 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 [98]). In order 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 generates 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 [36], [114]. 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 [74], [214], [215] (*i.e.,* unlabeled positives), because they are top-ranking retrieval results of high relevance scores to queries. To resolve this issue, RocketQA [36] 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 [216] 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.

5.2.4 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 5.2.1) cannot generate sufficiently informative negatives for dense retrieval [98], [104], [204]. In particular, Lu et al. [104] 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 [98]: 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 [98] 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. [111] 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.

In machine learning, negative sampling is a commonly adopted learning strategy for optimizing over the large candidate space [217], [218], which has been extensively studied in the literature. Here, we limit the scope of our discussion to dense retrieval.

5.3 Data Augmentation

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. In this section, we discuss two major approaches for data augmentation: the former incorporates additional labeled datasets, while the latter generates pseudo relevance labels by knowledge distillation.

5.3.1 Auxiliary Labeled Datasets

Several studies propose to incorporate auxiliary labeled datasets for enriching the relevance judgements. Karpukhin et al. [1] collect five datasets of NQ [33], TriviaQA [56], WebQuestions [58], TREC [59] and SQuAD [55], 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 [1], 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. [219] 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. [107] 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 [107], 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]–[222] (detailed in Section 8.1).

5.3.2 Knowledge Distillation

Considering that human-generated 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*) [223]. 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 [36] 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 [110]. Further, an empirical study [101] 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. [103] explore the possibility of using an enhanced bi-encoder (*i.e.,* ColBERT [95] that uses late interaction) as the teacher network. As another interesting work, Yu et al. [224] utilize a well-trained ad-hoc dense retriever to improve the query encoder in a conversational dense retriever, in order 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 [36] 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:

$$\mathcal{L}^{KD}_{MSE} = \frac{1}{2}\sum_{q\in\mathcal{Q}}\sum_{d\in\mathcal{D}}(r^{(t)}_{q,d} - r^{(s)}_{q,d})^2, \tag{17}$$

  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:

$$\mathcal{L}^{KD}_{KL} = -\sum_{q\in\mathcal{Q},d\in\mathcal{D}}\tilde{r}^{(s)}_{q,d}\cdot\big(\log\tilde{r}^{(s)}_{q,d} - \log\tilde{r}^{(t)}_{q,d}\big). \tag{18}$$

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

$$\mathcal{L}^{KD}_{MM} = \sum_{q,d_1,d_2}\max\Big(0, \gamma - \text{sign}(\Delta^{(t)}_{q,d_1,d_2}\times\Delta^{(s)}_{q,d_1,d_2})\Big), \tag{19}$$

  where $q\in\mathcal{Q}$, $d_1,d_2\in\mathcal{D}$, $\Delta^{(t)}_{q,d_1,d_2} = r^{(t)}_{q,d_1} - r^{(t)}_{q,d_2}$, $\Delta^{(s)}_{q,d_1,d_2} = r^{(s)}_{q,d_1} - r^{(s)}_{q,d_2}$, γ 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:

$$\mathcal{L}^{KD}_{M-MSE} = \text{MSE}(r^{(t)}_{q,d^+} - r^{(t)}_{q,d^-}, r^{(s)}_{q,d^+} - r^{(s)}_{q,d^-}). \quad (20)$$

To examine the effectiveness of different distillation functions, Izacard et al. [105] 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 [101], [110], *e.g.,* pointwise MSE loss.

Advanced distillation methods. Recent studies [225], [226] 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 [140], [141], [154] 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, in order to fit the learning of the student model. PROD [154] proposes to use a progressive chain of the teacher model with improved model architectures and increased network layers: 12-layer bi-encoder $\rightarrow$ 12-layer cross-encoder $\rightarrow$ 24-layer cross-encoder (given the 6-layer bi-encoder as the student model). ERNIE-Search [141] 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 [140] 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 [38] 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 (18)), the teacher model is also able to be adjusted according to the student model, yielding a better distillation performance.

The above two approaches can directly produce pseudo supervision signals for *fine-tuning* the underlying PLMs for dense retrieval. Besides, there are also other augmentation related methods for dense retrieval, such as synthetic data generation [116], [227], [228] and contrastive learning [132], [229], [230], which are more related to *pretraining*. We leave the discussion of these topics in Section 5.4.

5.4 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 sub-optimal performance on downstream applications due to the lack of task-specific optimization [92], [93], [127], [211]. 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)[11]. Next, we describe the pretraining approaches for dense retrieval in detail.

5.4.1 Task Adaptive Pretraining

This line of pretraining tasks essentially follow what have been used in BERT [28], 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) [92]: ICT 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) [93]: BFS utilizes the sentences from the first section of a Wikipedia article as anchors to the rest sections of texts. It considers a randomly sampled 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) [93]: WLP 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 page of the query is considered text.
- Recurring Span Retrieval (RSR) [127]: RSR 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) [211]: ROP 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* [231] 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.

5.4.2 Generation-Augmented Pretraining

Although the above pretraining tasks can improve the retrieval performance to some extent, they still rely on self-supervised signals that are derived from original text. Considering this issue, several studies propose to directly

11. Note that some of the techniques presented in this part are also related to data augmentation. We discuss these techniques in this section, because they are more focused on the pretraining stage instead of the fine-tuning stage.

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. [227] 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. [228] 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 [116] pretrains the bi-encoder retrievers, leading to consistent performance improvement over the variants pretrained with the tasks of ICT and BFS (Section 5.4.1).

Instead of using a pipeline way, Shakeri et al. [232] propose an end-to-end approach to generating question-answer pairs based on machine reading comprehension by using a pretrained LM (*e.g.,* BART [233]). 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. [234] further extend this approach by incorporating an additional selection step for enhancing the generated question-text pairs for retrieval tasks.

Furthermore, several studies [99], [126], [234], [235] also explore the generation-augmented pretraining approach in the *zero-shot setting*, where there are no training datasets in the target domain. Their results show that generation-augmented pretraining is useful to improve the zero-shot retrieval capacity of dense retrievers. We will discuss more on zero-shot retrieval in Section 8.1.

5.4.3 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 [94] utilizes a knowledge retriever for retrieving relevant texts from a large background 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. [236] 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 question answering systems.

This part is also related to the topic of retrieval-augmented language model, which will be discussed in Section 8.4.

5.4.4 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 sub-optimal 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 [237]–[240], 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. [112] 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 TSDAE [90], which is a Transformer-based sequential denoising autoencoder for enhancing the text representation. Furthermore, co-Condenser [78] 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 [241][12], recent studies [145], [152] 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. [124] 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

12. In [241], 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".

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 [145] 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 [142] 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. [242] 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 [152] applies such a representation enhanced pre-training strategy to learned sparse retrieval (*e.g.,* SPLADE [172]) based on PLMs, where it incorporates lexicon-based representation bottleneck (*i.e.,* continuous bag-of-words representations with learned term weights) for pretraining.

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 [243] and MoCo [244]), 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 [229] produces positives of one sample text by applying different dropout masks, and uses in-batch negatives. ConSERT [230] uses four ways to generate different views (*i.e.,* positives) for texts, including adversarial attacks, token shuffling, cutoff and dropout. Contriever [130] 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 [130]. In a similar manner, LaPraDoR [132] 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. [139] 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 [245] (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 (Section 5.4.1, 5.4.2 and 5.4.3), 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 [78], [90], [124], [132], [145]. Besides, it has been shown that these approaches can improve the retrieval performance even in zero-shot or low resource settings [112], [130], [132], which also alleviates the scarcity issue of relevance judgements.

5.5 Empirical Performance Analysis with RocketQA

Previous sections have extensively discussed various optimization techniques to improve the training of dense retrievers. In this section, we take RocketQA [36] as the studied model and examine how different optimization techniques improve its retrieval performance on benchmark datasets.

5.5.1 Experimental Setup

To prepare our experiments, we adopt the widely used MS MARCO passage retrieval dataset [32] for evaluation, and the detailed statistics of this dataset are reported in Table 1.

For comparison, we consider two variants of RocketQA, namely RocketQA [36] and RocketQAv2 [38], and a related extension called RocketQA$_{PAIR}$ [37][13]. For RocketQA, it uses three major training tricks, namely cross-batch negatives (Section 5.2.2), denoised hard negatives (Section 5.2.3) and distillation-based data augmentation (Section 5.3.2). For RocketQAv2, it incorporates a new soft-label distillation mechanism, called dynamic listwise distillation (Section 5.3.2). For RocketQA$_{PAIR}$, it is built on RocketQA and incorporates the passage-centric pretraining technique (Section 5.1.1). Considering the tunable configuration (*e.g.,* the number of negatives and batch size) of these models, we include multiple comparisons with different settings.

Although RocketQA variants do not include all the optimization techniques introduced in Section 5, they provide a unified base model by examining the effects of different

13. It was originally called *PAIR*, and we call it RocketQA$_{PAIR}$ for the consistency of the naming.

optimization techniques in a relatively fair way. As a comparison, we incorporate the classic BM25 [16] and DPR [1] 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 4. This code repertory is implemented based on the library PaddlePaddele, consisting of RocketQA, RocketQA$_{PAIR}$ and RocketQAv2. For a fair comparison, we re-implement the DPR model with the library PaddlePaddele [246] and enhance it with the ERNIE model [176] as the base PLM[14]. For simplicity, we only show the key commands or code to reproduce these comparison results in Table 5.

5.5.2 Results and Analysis

Table 4 presents the performance comparsion of different methods or variants on the MS MARCO dataset for the passage retrieval task. We have the following observations:

• Firstly, cross-batch technique is able to improve the retrieval performance. Compared with 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 *v.s.* $2a$). Besides, we can see that 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.

• Secondly, 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 $2a$ *v.s.* $2i$). 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.

• Thirdly, 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 $2i$ *v.s.* $2l$). Besides, as we can see, the performance improves with the increasing amount of pseudo labeled data.

• Fourthly, passage-centric pretraining can boost the performance, which leads to a substantial improvement of 0.7 percentage point in absolute $MRR@10$ performance (variant $3a$ *v.s.* $3b$). This technique characterizes a more comprehensive semantic relation among query, positive texts, and negative texts.

• Finally, the variant with dynamic listwise distillation (variant $4e$) achieves the best performance among all the RocketQA variants. The dynamic listwise distillation technique provides a joint training approach for retriever and reranker based on soft-label distillation (detailed in Section 7.2.3).

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

14. ERNIE [176] is a knowledge enhanced PLM by incorporating knowledge bases into text-based PLMs for pretraining.

TABLE 4
Empirical comparison of different RocketQA variants on MS MARCO dataset for the passage retrieval task (in percentage). CB=cross-batch, IB=in-batch, HLD=hard-label distillation, SLD=soft-label distillation, and DeHN = denoised hard negatives.

Model	Technique		MRR@10
BM25	(0)	BM25	18.7
DPR$_{ERNIE}$	(1)	in-batch ($batchsize = 4096$)	32.4
RocketQA	(2a)	cross-batch ($batchsize = 4096$)	33.3
	(2b)	cross-batch ($batchsize = 2048$)	32.9
	(2c)	cross-batch ($batchsize = 1024$)	31.4
	(2d)	cross-batch ($batchsize = 512$)	29.8
	(2e)	cross-batch ($batchsize = 256$)	29.5
	(2f)	cross-batch ($batchsize = 128$)	28.6
	(2g)	cross-batch ($batchsize = 64$)	26.9
	(2h)	CB + hard neg w/o denoising	26.0
	(2i)	CB + hard neg w/ denoising	36.4
	(2j)	CB + DeHN + data aug. (208K)	36.5
	(2k)	CB + DeHN + data aug. (416K)	36.9
	(2l)	CB + DeHN + data aug. (832K)	37.0
RocketQA$_{PAIR}$	(3a)	IB + hard-label distillation	37.2
	(3b)	IB + HLD + p-centric pretraining	37.9
RocketQAv2	(4a)	soft-label distillation	36.5
	(4b)	SLD + joint training (7 HNs)	36.5
	(4c)	SLD + joint training (31 HNs)	37.3
	(4d)	SLD + joint training (63 HNs)	38.1
	(4e)	SLD + joint training (127 HNs)	38.3

6 Indexing for Dense Retrieval

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

6.1 Traditional Inverted Index for Sparse Retrieval

To start with, we review the key data structure, *i.e.,* term-based inverted index, to support traditional sparse retrieval.

In essence, inverted index maintains an efficient mapping from terms to their locations in documents [247]. 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 basic idea of using the inverted index for relevance evaluation is to search the documents by query terms and evaluate 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 sparse retrieval toolkits, Apache Lucene[15] and its extensions including Elastic Search[16] and Apache Solr[17] have become *de facto* software for building inverted index based retrieval systems. Besides, Anserini [83], also built on

15. https://lucene.apache.org
16. https://elastic.co
17. https://solr.apache.org

TABLE 5
Important command parameters to implement different dense retrieval models in our toolkit.

Parameter name	Value	Setting	Related model
use_cross_batch	false true	In-batch negative Cross-batch negative	RocketQA
batch_size	number	Batch size per GPU	RocketQA, PAIR, RocketQAv2
train_set	file_name	Training data for different strategies (DeHN, data aug., etc.)	RocketQA, PAIR, RocketQAv2
is_pretrain	true false	Incorporating passage-centric pretraining Query-centric fine-tuning	PAIR

top of Lucene, allows researchers to easily reproduce sparse retrieval models on standard IR test collections.

6.2 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 existing literature of dense retrieval, most of previous studies adopt the Faiss library [84] to implement nearest neighbor search, while lacking a detailed discussion about the underlying data structure and implementation algorithms. In practice, nearest neighbor search is a crucial technique in order to design efficient dense retrieval systems. Additionally, there exists some connection between the two research communities of *information retrieval* and the *nearest neighbor search*. Thus, it is worthwhile to have a discussion of nearest neighbor search in this survey. 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.

6.2.1 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 text 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 [248], [249] 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 [248]–[251] 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. In order to evaluate different ANNS aglorithms, ANN-Benchmarks [252][18] has been released with a million-scale benchmark, and an enlarged version called Big-ANN-Benchmarks [253][19] 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 6.2.2) and product quantization algorithms for ANNS (Section 6.2.3), and also introduce publicly available software of ANNS for building efficient dense retrieval systems (Section 6.3).

6.2.2 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* [248], [254], [255] 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* [256] 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 are likely to 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-

18. http://ann-benchmarks.com/
19. https://big-ann-benchmarks.com/

scale data, where cluster centroids are stored in memory while posting lists to centroids are stored in disk [257].

• *Graph-based approaches* [258], [259] (*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 searches on the graph by greedily selecting the similar neighbors. Such approaches are efficient due to the *small world phenomenon* [260]: the average path length of between two vertices is short. Graph-based approaches usually perform well on large high-dimensional datasets, while taking a high memory cost.

6.2.3 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 (*i.e.,* text embeddings) are composed of real-value numbers, and it takes significantly more time for embedding similarity computation, also leading to an increase in memory cost.

In this part, we describe a *product quantization* (PQ) approach [261], [262] 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 [261], [262] 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 the 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. In order 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: it only requires a total of $b_1 \cdot b_2$ embedding similarity computes, independent of the corpus size. With this similarity table, the similarity evaluation of a text can be efficiently implemented by table lookup.

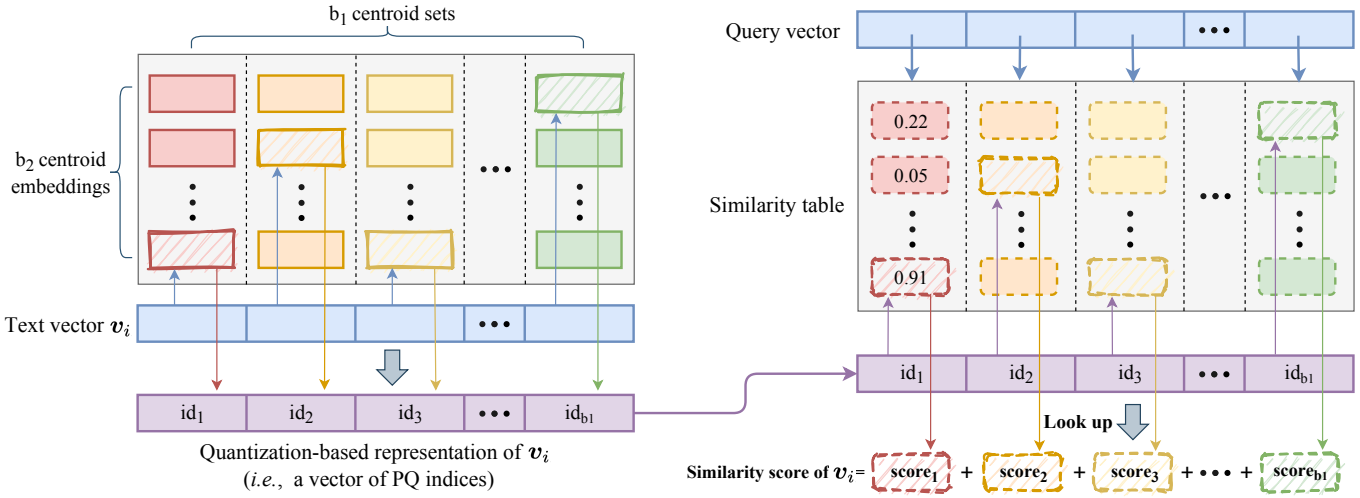
Optimizing quantization based index for retrieval. The original purpose of product quantization is to solve general vector compression problems, and it cannot be directly end-to-end learned in neural network models, due to the involved non-differentiable operations. Besides, dense retrieval also requires specific optimization strategies (*e.g.,* negative sampling), which are directly applicable to product quantization techniques. To address the above issues, Zhan et al. [117] 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 this work [117], the authors further incorporate the technique of constrained clustering to learn a better centriod assignment for a given document [122]. Besides, Zhang et al. [263] also propose several improved optimization strategies (including gradient straight-through estimators, warm start centroids and Givens rotation) to jointly train deep retrieval models with quantization based embedding index. Yamada et al. [264] 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 [265].

6.3 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 [98] or retrieve supporting context [94].

As aforementioned, existing dense retrieval studies mainly adopt Faiss [84] for implementing nearest neighbor search. Faiss is a publicly released library for efficiently searching and clustering large-scale dense vectors. Basically, it provides the exact nearest neighbor search function, and further supports the implementations of several ANNS approaches discussed in Section 6.2 (including clustering-based approaches, graph based approaches, and quantization approaches) in both CPU implementations and GPU implementations. It also supports different similarity functions for vector search, including dot product and cosine similarity. In order to meet the requirements of effectiveness and efficiency, users can set the appropriate configuration



(a) Quantization-based representation (b) Accelerated similarity computation

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 assigned with 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.

with Faiss according to different hardware profile and data sizes.

Besides, HNSWlib [266] and SPTAG [267] are libraries focusing on the efficient implementation for graph-based approach. ScaNN [268] is developed with quantization-based approaches, which achieves excellent performance on ANN-Benchmarks. Distributed-Faiss [269] provides the distributed solutions for ANNS, when the index is too large to fit into the memory of a single machine.

Besides the above open-sourced software, Pinecone[20] and Google's Vertex AI Matching Engine[21] provide commercial service for ANNS algorithms, which can help developers easily conduct dense retrieval systems for applications. Note that this survey mainly limits the discussion to in-memory solutions. When the data size is extremely large (*e.g.*, 100 billion scale), it is not feasible to use completely in-memory ANNS index for dense retrieval, and we need to develop multi-level index using a hybrid of memory index and disk index, *e.g.*, DiskANN [270] and SPANN [257].

7 Integration with Reranking

In a complete retrieval system, it usually consists of first-stage retrievers and subsequent rerankers. In what follows, we first briefly introduce the retrieval pipeline, then present several approaches to optimizing the retrieval pipeline, and finally extend the discussion to the use of dense retrievers in other application systems.

7.1 The Retrieval Pipeline

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

20. https://pinecone.io/
21. https://cloud.google.com/vertex-ai

7.1.1 The General Retrieval Pipeline

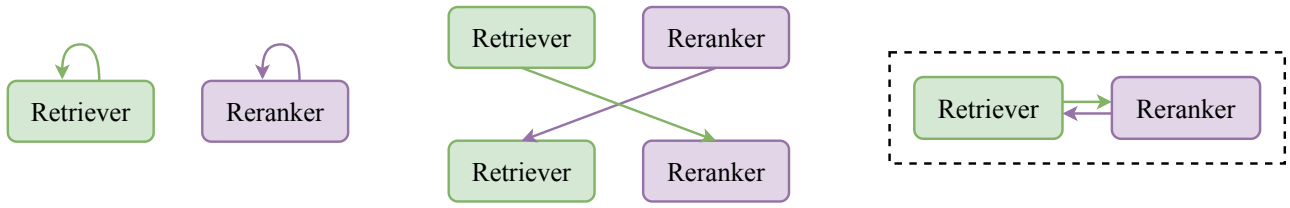
To start our discussion, we consider a simplified retrieval pipeline, consisting of two major stages, namely the first-stage retrieval and the reranking. To obtain the final ranked list, a certain number of possibly relevant documents (*e.g.*, several hundreds to thousands) to a given query are retrieved 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 [25], [193], [271]. 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 [25], [26], [271] for detailed discussions.

7.1.2 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.



(a) Separate training. (b) Adaptive training. (c) Joint training.

Fig. 4. Comparison of the three approaches for pipeline training of retriever and reranker.

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 [193], [272]–[274]. Specifically, BERT is widely used to implement the cross-encoder for estimating the relevance score [193], [272], *e.g.*, monoBERT [275]. 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 (13)). 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 [193] using the binary cross-entropy (BCE) loss as shown in Equation (14). 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 [276], [277]. Furthermore, duoBERT [275] 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 [41] or ranking score [278].

Multi-stage ranking. To construct a full system, a commonly used strategy is to arrange the retriever and the reranker(s) in a processing pipeline. Nogueira et al. [275] present a multi-stage ranking architecture in order 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 [279] for multi-stage ranking optimization [22]. 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 [281] and Walmart Product Search [282]), where more comprehensive factors are considered and more complicated strategies are designed.

Besides the above work, there is a large body of studies that utilize PLMs to enhance pre-BERT neural ranking models, *e.g.*, CEDR [283]. 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 [29], [93], [284].

7.2 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 as shown in Figure 4. For simplicity, we only discuss the scenario where only a retriever and a reranker are involved in a retrieval pipeline.

7.2.1 *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) between the two components. Tyically, we can firstly optimize the retriever and then learn the reranker, following the optimization methods in Section 5 and Section 7.1, respectively.

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 [36], [277]. 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

22. A similar approach SetRank [280] has been proposed, though it does not use PLMs.

entire pipeline [38]. Therefore, it is necessary to incorporate information interaction between the retriever and reranker during training.

7.2.2 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 in order to achieve a better overall retrieval performance. For example, Qu et al. [36] 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 [120] 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 [121]: 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.

7.2.3 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*[23]. 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 [38] 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.

Besides, Sachan et al. [108] present an end-to-end training method to jointly optimize the retriever and the reader (similar methods can apply to reranker) for retrieval-augmented question-answering systems. They propose two training approaches by modeling the retrieved documents, either *individually* or *jointly*.

7.3 Integration and Optimization in Other Applications

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. Typically, these application systems contain two major parts, namely knowledge retriever (the component provides necessary supporting evidences) and task solver (the component reasons over the retrieved evidences for prediction). Compared with traditional sparse retrievers, dense retrievers are more flexible to be integrated into downstream tasks. It is conceptually simple, and can be effectively optimized according to specific task goals.

To optimize a retrieval-augmented application system, a typical approach [94] is to construct an embedding index for knowledge retriever and then utilize the retrieved text embeddings for optimizing the task solver. During the training of the whole system, a major difficulty lies in the issue of *index staleness*: at each training step, when we update the knowledge encoder, the indexed embeddings no longer correspond to the latest parameters of the trainer. To address this issue, REALM [94] proposes to use *asynchronous index update*: the index builder is periodically (*e.g.*, each for several hundred training steps) updated, and the trainer utilizes an existing index for optimization before it is reconstructed[24]. Such an approach can be generally applied to optimize downstream systems built on dense retrievers. Other related studies [108], [286], [287] 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 9.

8 Advanced Topics

In this section, we discuss several important advanced topics for dense retrieval, including zero-shot dense retrieval, model robustness to query variations, model based retrieval and retrieval-augmented language model.

8.1 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. Thus, it is important to examine the

23. Following [38], *listwise* denotes that the optimization is based on an entire candidate list, which is different from that in learning to rank [285].

24. The asynchronous update strategy can be utilized in other optimization tasks that rely on indexed embeddings, *e.g.*, hard negative retrieval [98].

zero-shot capabilities of dense retrievers, as well as their out-of-distribution performance.

Zero-shot evaluation datasets. To our knowledge, BEIR [72] is the first heterogeneous benchmark for examining the zero-shot capabilities of dense retrieval methods. BEIR includes nine different retrieval tasks (news retrieval, question answering, bio-medical information retrieval, etc.) 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. [52] 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. [194] 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.

Empirical analysis of the influence factors. 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 between sparse and dense retrieval models: since some datasets are constructed based on lexical matching models [47], [61], they tend to be more favorable for sparse retrieval methods due to a larger term overlap.

Exisiting solutions for zero-shot retrieval. Recently, it has attracted much attention from the research community to improve the zero-shot capabilities of dense retrievers. Following [220], we briefly summarize these studies and discuss how they address the issues for zero-shot retrieval in three major aspects.

• *Augmenting the target training data*. For zero-shot retrieval, the major challenge is that it lacks the training data from the target domain. Considering this issue, 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 [99], [126], [234], [235]. Recently, Promptagator [153] leverages a large language model consisting of 137 billion parameters, *i.e.,* FLAN [288], to generate large-scale query-text pairs by using only a small number of labeled examples. It trains a dense retriever with the set of generated query-text pairs, and shows that the retriever trained on the generated data (with consistency filtering) significantly outperforms ColBERTv2 [125] that is well-trained on manually annotated dataset (*e.g.,* MS MARCO) on BEIR. 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 [102], [123], [126] in zero-shot retrieval scenarios. Besides, several studies [130], [132] conduct unsupervised pretraining by leveraging large-scale positive and negative pairs with different data augmentation methods, *e.g.,* ICT [92] and SimCSE [229].

• *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 [72], [289], 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 [289] or relevance scores [132] for both sparse and dense retrievers. Besides, we can also employ knowledge distillation for improving dense retrievers [123], taking sparse retrievers as the teacher model.

• *Scaling up the model capacity.* In recent years, scaling law for PLMs 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. In [129], 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 [141].

Besides, it is also useful to employ multi-task learning (*e.g.,* a joint training with self-supervised tasks, dense retrieval and extractive question answering [290]) to improve the zero-shot retrieval performance. As another related study, Zhan et al. [221] 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 [222].

8.2 Improving the Robustness to Query Variations

Besides the out-of-distribution issue in zero-shot retrieval, it has been shown that dense retrieval models are more sensitive to *query variations* than traditional lexical matching methods [146], [291], [292]. 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. [291] 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. [292] aim to examine how different types of query variations negatively affect the robustness of the retrieval pipeline. In order 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 [146], 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. In order 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. [291] 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 [293] 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 [291], the same authors [137] 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. [146] 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.

While, query variations are not always harmful for retrieval systems. For example, we can utilize *query extension* or *query rewriting* [294] to derive better search results by reformulating the issued query. Furthermore, there are several studies that focus on the vulnerabilities of dense retrieval models [295], [296]. It has been shown that dense retrieval models are brittle to deliberate attacks and thus adversarial training approaches are often used for enhancing the model robustness [295], [296]. While, we limit our discussion to query variations, which are more likely to occur in real search scenarios than deliberate attack [292].

8.3 Model based Retrieval

In recent years, *model based retrieval* [297], [298] has been proposed as an alternative paradigm for information retrieval. During retrieval, it no longer examines the index storing the candidate texts in the collection, but instead directly predicts the identifiers of relevant documents (*i.e.,* docids) based on a parametric model. The underlying model is expected to fully capture the necessary relevance information, which is the key to this retrieval approach.

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 [299], where entity identifiers (*e.g.,* unique entity names) are generated conditioned on the text context in an autoregressive manner, and then GRLS [300] extends this idea to long sequence retrieval by introducing multi-step generative retrieval. Specially, the perspective paper [297] envisions the *model-based paradigm* that simplifies the classic *index-retrieve-then-rank* paradigm by a unified relevance model. Further, a representative work [298] 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 [298] 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 compares the three docid assignment methods, and shows that the third method leads to the best performance in its implementation. Later extensions almost follow the three major methods for representing docids: DynamicRetriever [301] (atomic docid: original docid), SEAL [302] (simple string docid: full ngram signatures with the efficient support of Ferragina Manzini index), NCI [303] (semantic string docids: hierarchical clustering), DSI-QG [304] (simple string docids: generated queries), Ultron [305] (simple string docids: URL and titles, semantic string docids: product quantization), GERE [306] (semantic string docids: title and evidence), and CorpusBrain [307] (semantic string docids: 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 [298], [305], [307], *e.g.,* mapping a sampled content (*e.g.,* passage, word sets and ngrams [301], [305]) or associated content (*e.g.,* anchor text [307]) 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 [304], [305], where the original content of a document can be employed to generate potential queries.

Merits. Compared with traditional retrieval approaches, model based retrieval potentially leads to a more unified retrieval paradigm. The major merits of this approach are threefold. Firstly, it simplifies the classic index-retrieve-then-rank paradigm by *a model-based paradigm*, where a more unified solution to information retrieval tasks could be developed (as envisioned in [297]). Due to the flexibility, there are increasing studies that utilize model based retrieval for solving different knowledge-intensive tasks [306], [307]. It can be potentially extended to index and retrieve various Web resources, *e.g.,* figures and videos. Secondly, since such an approach basically takes an encoder-decoder architecture, it is more easier to be optimized in an end-to-end way. We could potentially get rid of the tedious pipeline training in existing retrieval systems. Thirdly, it directly removes the use of elaborate inverted-index or large-sized embedding index. During retrieval, we do not need to manage a complicated index structure for retrieval, which can largely simplify the data structure for supporting the retrieval system.

Despite these merits (unified paradigm, training flexibility and simplified structure), this emerging retrieval paradigm is still under exploration. So far, 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.,* the full MS MARCO dataset) is still challenging for model based retrieval.

8.4 Retrieval-Augmented Language Model

In recent years, neural language models (LM), especially PLMs, have proven to be powerful [308]–[311] in a variety of tasks. It is shown that PLMs can encode a large amount of semantic knowledge [161], [312] into large-scale model parameters. In order to enhance the representation capacity, a number of studies explore the scaling law of PLMs [161], [165] by increasing the model size, which makes the training and use of PLMs prohibitively expensive in a resource-limited setting.

In order to alleviate this issue, retrieval-augmented LMs have been proposed by allowing the model to explicitly access external data [94], [313]–[316]. 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. [313] present kNN-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 kNN-LM works particularly well when predicting rare patterns, since it allows explicit retrieval of rare patterns. Furthermore, Yogatama et al. [315] incorporate a gating mechanism to replace the fixed interpolation parameter in kNN-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. [316] 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. [94] propose REALM to jointly optimize the knowledge retriever and the knowledge-augmented encoder (detailed in Section 5.4.3).

9 Applications

As an important information seeking technique, dense retrieval has been widely applied in a variety of applications from different fields. In this section, we review the applications of dense retrieval technique in three aspects, namely the applications to information retrieval tasks (*extensions to different retrieval settings*), the applications to natural language processing tasks (*benefiting downstream tasks*) and industry practice (*industry-level use and adaptation*).

9.1 Information Retrieval Applications

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

9.1.1 Temporal Retrieval

Most of existing works conduct the study of dense retrieval on synchronic document collections (*e.g.,* Wikipedia). However, there is a large proportion of temporal-dependent questions in real application scenarios. To advance the research on temporal retrieval, Zhang et al. [317] create an open-retrieval question answering dataset called *SituatedQA*, where the systems are required to answer questions given the tempo-spatial contexts. Their experiments show that existing dense retrievers cannot produce satisfying answers that are frequently updated. Besides, Wang et al. [64] create another large question answering dataset corpus called *ArchivalQA*, consisting of 1,067,056 question-answer pairs, which is specially constructed for temporal news question answering. These datasets can be used to develop time-sensitive dense retrievers.

9.1.2 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.,* lists, tables and knowledge bases. Herzig et al. [318] 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 [318], the proposed approach significantly outperforms BM25 on the NQ-TABLES dataset. Oguz et al. [107] 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. [319] also explore the unified retrieval by proposing a tri-encoder, with one unique encoder each for question, text and table, respectively. Their experimental results show that the tri-encoder performs better than the bi-encoder with one encoder for the question, the other one for both text and tables.

9.1.3 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. [44] 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. [320] present the multi-lingual benchmark dataset called *Mr. TYDI*, consisting of monolingual retrieval corpus in eleven langauges. 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. [321] 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. [322] 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 [323]), and achieves performance improvement in a number of languages.

9.1.4 Other Retrieval Tasks

So far, dense retrieval has been widely applied to various domain-specific retrieval tasks, such as mathematical IR [324], code retrieval [325] and biomedical IR [326]. 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 [327], 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 [328], 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 [327] for a detailed discussion.

9.2 Natural Language Processing Applications

Many NLP tasks need to retrieve relevant supporting evidence or knowledge from external sources, *e.g.,* text corpus, knowledge graph and tables. In this section, we will discuss the applications of dense retrieval to three typical NLP tasks, including question answering, entity linking and dialog system.

9.2.1 Question Answering

Open-domain question answering (QA) is an information seeking task that aims to find accurate answers to a question instead of relevant texts. A QA system typically conducts a two-stage approach [39], where a text retriever is firstly employed to find relevant documents (or passages) from a large text corpus and then another component (called *reader*) derives the exact answer from these documents.

Traditionally, the text retriever in QA systems is often implemented by sparse retrieval methods such as BM25

and tf-idf. As an important progress, ORQA [92] (for the first time) demonstrates that dense retrieval models can outperform BM25 on multiple open-domain QA datasets. It pretrains the retriever with an Inverse Cloze Task in a self-supervised manner (as discussed in Section 5.4), and then jointly fine-tunes the retriever and reader. However, ORQA needs computationally intensive pretraining. By fine-tuning the BERT model only on annotated question-passage pairs, DPR [1] no longer requires additional pretraining, while the experiments demonstrate that DPR performs even better than ORQA on multiple QA datasets. Moreover, RocketQA [36] proposes an optimized training approach to dense retrieval, leading to a better performance on QA tasks. These studies rely on extractive readers to obtain answers from retrieved documents. Different from these studies, RAG [287] and Fusion-in-Decoder [286] use seq2seq based generative models for predicting the answer, showing a better performance on open-domain question answering than extractive readers.

The above studies focus on the simple setting where the answers can be derived from a single supporting document, *i.e.,* the single-hop setting. While, in a more complicated setting, the QA system needs to collect sufficient evidence from multiple documents for finding the answer, which requires the capacity of multi-hop reasoning (*e.g.,* HotpotQA [57] and HoVer [329]). In early studies [330], hyperlink structure in the corpus has been utilized to thread relevant documents for locating the answer. More recently, without using hyperlinks, *multi-hop dense retrieval* [331], [332] has been proposed to enhance the performance of complex QA tasks [57]. Specifically, it iteratively encodes the combination of the question and the retrieved documents as a new question embedding, and then retrieves the documents relevant to the new query for the next hop. Experiment results show that these approaches work well for two-hop tasks. However, the many-hop retrieval task remains challenging, as each hop is likely to propagate errors. To better deal with many-hop (more than two hops) tasks, Khattab et al. [331] further propose the *condensed retrieval* architecture, which enhances the query informativeness by extracting the facts from each hop as part of the query for the next hop.

More recently, the survey paper [333] extensively discusses the recent progress on open domain question answering based on dense retrieval. The readers can refer to this survey for a detailed introduction.

9.2.2 Entity Linking

Entity linking is a semantic disambiguation task that relates specific contextual references to real-world entities. Given a text context containing a specific mention, the entity linking system aims to link it to a corresponding entity in a knowledge base. Since knowledge bases are usually large (*e.g.,* English Wikipedia contains more than six million articles [25]), a major challenge of entity linking is to efficiently retrieve a small number of candidate entities for linkage. Traditional approaches mainly use a high-coverage alias table or sparse retrieval methods (*e.g.,* TF-IDF and BM25) for efficient candidate retrieval, and further employ neural models for effective candidate ranking [334]–[336].

Based on dense retrieval, researchers use PLMs for candidate recall or reranking in entity linking. Logeswaran et al. [337] explore BERT-based rankers with different architectures (*e.g.,* bi-encoder and cross-encoder) for candidate ranking but rely on traditional sparse retrieval for candidate recall. As neural two-tower architecture (pre-BERT models) is shown effective for candidate recall [5], Wu et al. [6] further propose a BERT-based two-stage approach for entity linking, namely BLINK. In its first stage, BLINK uses a BERT-based bi-encoder to retrieve candidates. The retrieved candidates are then reranked by a BERT-based cross-encoder. In BLINK, when training the bi-encoder, hard negative mining and knowledge distillation are also used to improve its retrieval performance.

9.2.3 Dialog System

Recent studies (*e.g.,* Meena [338], BlenderBot [339], and PLATO-2 [340]) that pretrain dialog models on large-scale corpora have demonstrated their ability to converse like a human to some extent. However, these models are still prone to suffering from factual incorrectness and knowledge hallucination in an open-domain setting [341].

Inspired by the success of Retrieval-Augmented Generation (RAG) in the task of open-domain question answering [287], the RAG architecture has been applied to open-domain knowledge-grounded dialog to address this issue [342], [343]. In this approach, a retriever trained in an end-to-end manner is used to access external knowledge sources based on dialog contexts. The retrieved knowledge is then used to enhance dialog generation. Experiment results show that these approaches can alleviate the hallucination issue without sacrificing the conversational ability [342].

Furthermore, several studies [341], [344] propose to generate a pseudo search query with generative models according to the dialog contexts. The generated query is then used to retrieve knowledge from a search engine or a pre-built information retrieval system. The retrieved knowledge and conversation history will be subsequently used to generate a knowledge-grounded response.

9.3 Industrial Practice

Besides research-oriented studies, considerable efforts have attempted to deploy dense retrieval techniques in real systems[26], including Web search [345], sponsor search [131], [346], product search [347], [348] and personalized search [281].

In Web search, the term mismatch between search queries and Web pages is one of the major challenges in practice. In order to enhance the semantic matching capacity, Liu et al. [345] 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 (as we discussed in

25. As of 25 November 2022, there are 6,579,781 articles in the English Wikipedia.

26. Due to the privacy of companies, we mainly survey the public industrial reports from the literature or blogging articles.

Section 4.3), an integration of dense retrieval and sparse retrieval techniques has been employed in Baidu search [345] and Spotify search[27].

In sponsor search [349], the systems should optimize multiple objectives simultaneously, *e.g.,* query-advertisement (ad) relevance and click-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 criterion. Fan et al. [346] argue that the separation of relevance learning and CTR prediction tends to produce suboptimal performance, and they propose to train a retriever that jointly considers both relevance and CTR. Specifically, they train the retriever from the synthetic clicked logs augmented by a teacher model. The proposed system is called *Mobius* and has been deployed into Baidu's sponsor search. As another solution, Zhang et al. [131] 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 learning 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 [131]. Besides, Dodla et al. [208] propose to integrate dense retrievers and non-autoregressive generators in a multi-task setting (sharing the same encoder), showing a significant performance gain by combining the two techniques in bid keyword retrieval.

In personalized search and product search, the personalized features and product features are important to consider in the retrievers. Huang et el. [281] incorporate the embeddings of search context (social relation, location, etc.) as the input of dense retriever. Liu et al. [348] incorporate the embeddings of product images as the input of dense retriever. Besides, Huang et al. [281] and Li et al. [347] also find that hard negatives are important for training dense retrievers (as we discussed in Section 5.2). Further, Magnani et al. [282] design a dense-sparse hybrid 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.

10 Conclusion

In this survey, we thoroughly review the recent progress of dense retrieval based on pretrained language models (PLM). 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.

27. https://engineering.atspotify.com/2022/03/introducing-natural-language-search-for-podcast-episodes/

PLM-based architecture. In dense retrieval, two kinds of PLM architectures have been extensively used, namely bi-encoder (*efficient* yet *less effective*) and cross-encoder (*effective* yet *less efficient*). In particular, bi-encoder is usually employed to implement the retriever and cross-encoder is usually employed to implement the reranker. Since the bi-encoder has limited capacity in capturing the semantic interaction between query and text, the multi-representation technique has been widely explored in enhancing the fine-grained interaction modeling. It proposes to compute the relevance score according to multiple contextual representations of a text. 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 [72]. Compared to sparse retrievers (*e.g.,* BM25), dense retrievers perform better in terms of semantic matching, but less well in terms of lexical matching [1], [52]. Thus, there has been increasing attention in combining the merits of both dense retrievers and sparse retrievers in a unified retrieval system [123], [147]. Moreover, dense retrieval models rely on latent semantic representations for relevance matching. It is not fully clear that how dense retrieval models behave and perform in response to different types of queries, *e.g.,* how they mimic "*bag-of-words*" retrieval [96]. More theoretical discussions are needed in order to better understand and enhance the retrieval capacity of dense retrieval models. Despite the rapid improvement, the problem whether dense retrieval can fully replace sparse retrieval is still being under explored by the research community, and there are also some studies that devise new framework consisting of multiple effectiveness measurements for examining this problem [350].

Training approach. Compared with previous neural IR methods, PLM-based dense retrieval models are more difficult to be optimized given the huge number of parameters. Even with the ever-increasing labeled dataset, it still requires specific training techniques to effectively optimize large retrieval models. We summarize 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, including negative selection, data augmentation and specific pretraining techniques. These studies have significantly improved the retrieval performance of dense retrievers on benchmark datasets [1], [36], [98], [112]. 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 static or dynamic hard negatives) typically lead to a better retrieval performance. Besides, in order 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 an improvement direction, it is important to design data- and parameter-efficient training approaches

for enhancing the retrieval capacity under low-resourced setting (*e.g.,* zero-shot retrieval).

Dense vector index. Although the idea of representing texts by dense vectors is intuitive, it is not easy to construct large-scale dense vector index for efficient search. Without the support of a suitable index structure, the retrieval process (*i.e.,* finding the most close text vectors *w.r.t.* query vectors) would be extremely slow. For this issue, a popular solution is the use of *Approximate Nearest Neighbor Search (ANNS)* algorithms [84], [248], [351], which can achieve highly efficient retrieval performance (*e.g.,* sublinear time cost [248]). Compared with term-based index, it is more difficult to maintain the ANNS index for dense retrieval, especially with the regular operations of adding, deleting and updating the embeddings of texts. For example, when the optimization involves in the update of indexed embeddings, it usually suffers from the *index staleness* issue [94]. Therefore, it is necessary to develop efficient online update algorithms for enhancing the flexibility of ANNS index[28]. In addition, dense vector index is typically used in memory, taking a higher cost compared to a term-based inverted index that can be stored on disk. Besides the effectiveness, the cost for maintenance and use is also important to consider for the practical deployment of dense retrieval systems [350]. Although quantization techniques can alleviate the cost issue to some extent, it is not easy to optimize quantization-based vector index due to the involved non-differential operation (with some joint optimization approaches proposed [117], [263]). Overall, the dense vector index should be well suited to the architecture and optimization of the PLM-based retrievers, which still requires more research 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). A preferred approach is to jointly optimize the retrieval pipeline, considering different components as a whole. 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 [275], 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 [38], [94]. 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 component to use for each of the stages) by learning from labeled dataset.

Besides the above four aspects, we have also extensively discussed several important advanced topics that have received increasing attention. In order to support this survey, we have created a referencing website (at the link https://github.com/RUCAIBox/DenseRetrieval) to provide useful resources on paper collection and code repertory for dense retrieval. This survey is expected to provide a systemic literature review that summarizes the major progress for PLM-based dense retrieval.

Coda: This survey was planned in June, 2021, when we drafted a preliminary outline to organize the content. We started the writing in September, 2021, and finished the initial version in December, 2021. Considering the rapid research progress of dense retrieval, we did not release the initial version online. In this year, we have largely revised this survey by extensively including recent papers and finally made it online, which we expect to be helpful for both researchers and engineers on the research of dense retrieval. It is not easy to write a long survey, and we will keep updating this survey and welcome any constructive comments to improve our survey.

28. There is commercial software (*e.g.,* https://pinecone.io) for maintaining the embedding index instantly (*i.e.,* add, edit or delete data instantly), while academic solutions are still lacking.

REFERENCES

[1] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, "Dense passage retrieval for open-domain question answering," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 6769–6781.

[2] E. M. Voorhees *et al.*, "The trec-8 question answering track report." in *Trec*, vol. 99, 1999, pp. 77–82.

[3] Z. Ji, Z. Lu, and H. Li, "An information retrieval approach to short text conversation," *CoRR*, vol. abs/1408.6988, 2014.

[4] H. Chen, X. Liu, D. Yin, and J. Tang, "A survey on dialogue systems: Recent advances and new frontiers," *SIGKDD Explor.*, vol. 19, no. 2, pp. 25–35, 2017.

[5] D. Gillick, S. Kulkarni, L. Lansing, A. Presta, J. Baldridge, E. Ie, and D. Garcia-Olano, "Learning dense representations for entity retrieval," in *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, 2019, pp. 528–537.

[6] L. Wu, F. Petroni, M. Josifoski, S. Riedel, and L. Zettlemoyer, "Scalable zero-shot entity linking with dense entity retrieval," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 6397–6407.

[7] B. Mitra, F. Diaz, and N. Craswell, "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 2017, Perth, Australia, April 3-7, 2017*, 2017, pp. 1291–1299.

[8] T. Joyce and R. Needham, "The thesaurus approach to information retrieval," *American Documentation (pre-1986)*, vol. 9, no. 3, p. 192, 1958.

[9] G. Salton, "Some experiments in the generation of word and document associations," in *Proceedings of the 1962 fall joint computer conference, AFIPS 1962 (Fall), Philadelphia, Pennsylvania, USA, December 4-6, 1962*, 1962, pp. 234–250.

[10] G. Salton, A. Wong, and C. Yang, "A vector space model for automatic indexing," *Commun. ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[11] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Inf. Process. Manag.*, vol. 24, pp. 513–523, 1988.

[12] A. Aizawa, "An information-theoretic perspective of tf–idf measures," *Information Processing & Management*, vol. 39, no. 1, pp. 45–65, 2003.

[13] S. Robertson, "Understanding inverse document frequency: on theoretical arguments for idf," *Journal of documentation*, 2004.

[14] J. Zobel and A. Moffat, "Inverted files for text search engines," *ACM Comput. Surv.*, vol. 38, p. 6, 2006.

[15] J. Zobel, A. Moffat, and K. Ramamohanarao, "Inverted files versus signature files for text indexing," *ACM Transactions on Database Systems (TODS)*, vol. 23, no. 4, pp. 453–490, 1998.

[16] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford *et al.*, "Okapi at trec-3," *Nist Special Publication Sp*, vol. 109, p. 109, 1995.

[17] S. Robertson and H. Zaragoza, *The probabilistic relevance framework: BM25 and beyond*, 2009.

[18] C. Zhai, *Statistical Language Models for Information Retrieval*, ser. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2008.

[19] T. Liu, "Learning to rank for information retrieval," in *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2010, Geneva, Switzerland, July 19-23, 2010*, 2010, p. 904.

[20] H. Li, "Learning to rank for information retrieval and natural language processing," *Synthesis Lectures on Human Language Technologies*, vol. 4, pp. 1–113, 2011.

[21] P. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. P. Heck, "Learning deep structured semantic models for web search using clickthrough data," in *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*, 2013, pp. 2333–2338.

[22] J. Guo, Y. Fan, Q. Ai, and W. B. Croft, "A deep relevance matching model for ad-hoc retrieval," in *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*, 2016, pp. 55–64.

[23] B. Mitra and N. Craswell, "Neural models for information retrieval," *arXiv preprint arXiv:1705.01509*, 2017.

[24] J. Guo, Y. Fan, L. Pang, L. Yang, Q. Ai, H. Zamani, C. Wu, W. B. Croft, and X. Cheng, "A deep look into neural ranking models for information retrieval," *Information Processing & Management*, vol. 57, no. 6, p. 102067, 2020.

[25] J. Lin, R. Nogueira, and A. Yates, "Pretrained transformers for text ranking: Bert and beyond," *Synthesis Lectures on Human Language Technologies*, vol. 14, no. 4, pp. 1–325, 2021.

[26] Y. Fan, X. Xie, Y. Cai, J. Chen, X. Ma, X. Li, R. Zhang, J. Guo, and Y. Liu, "Pre-training methods in information retrieval," *arXiv preprint arXiv:2111.13853*, 2021.

[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 2017, pp. 5998–6008.

[28] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 4171–4186.

[29] Y. Cai, Y. Fan, J. Guo, F. Sun, R. Zhang, and X. Cheng, "Semantic models for the first-stage retrieval: A comprehensive review," *arXiv preprint arXiv:2103.04831*, 2021.

[30] N. Craswell, B. Mitra, E. Yilmaz, D. Campos, and J. Lin, "Overview of the trec 2021 deep learning track," in *Text REtrieval Conference (TREC)*, 2022.

[31] N. Craswell, B. Mitra, E. Yilmaz, and D. Campos, "Overview of the TREC 2020 deep learning track," *arXiv preprint arXiv:2102.07662*, 2021.

[32] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng, "MS MARCO: A human generated machine reading comprehension dataset," in *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016*, ser. CEUR Workshop Proceedings, vol. 1773, 2016.

[33] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, K. Toutanova, L. Jones, M. Kelcey, M.-W. Chang, A. M. Dai, J. Uszkoreit, Q. Le, and S. Petrov, "Natural questions: A benchmark for question answering research," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 452–466, 2019.

[34] N. Craswell, B. Mitra, E. Yilmaz, D. Campos, and E. M. Voorhees, "Overview of the trec 2019 deep learning track," *arXiv preprint arXiv:2003.07820*, 2020.

[35] N. Tonellotto, "Lecture notes on neural information retrieval," *CoRR*, vol. abs/2207.13443, 2022.

[36] Y. Qu, Y. Ding, J. Liu, K. Liu, R. Ren, W. X. Zhao, D. Dong, H. Wu, and H. Wang, "RocketQA: An optimized training approach to dense passage retrieval for open-domain question answering," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021, pp. 5835–5847.

[37] R. Ren, S. Lv, Y. Qu, J. Liu, W. X. Zhao, Q. She, H. Wu, H. Wang, and J.-R. Wen, "PAIR: Leveraging passage-centric similarity relation for improving dense passage retrieval," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 2021, pp. 2173–2183.

[38] R. Ren, Y. Qu, J. Liu, W. X. Zhao, Q. She, H. Wu, H. Wang, and J.-R. Wen, "Rocketqav2: A joint training method for dense passage retrieval and passage re-ranking," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 2825–2835.

[39] D. Chen, A. Fisch, J. Weston, and A. Bordes, "Reading Wikipedia to answer open-domain questions," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 1870–1879.

[40] J. Lin, "A proposed conceptual framework for a representational approach to information retrieval," *SIGIR Forum*, vol. 55, no. 2, pp. 4:1–4:29, 2021.

[41] R. Nogueira, Z. Jiang, R. Pradeep, and J. Lin, "Document ranking with a pretrained sequence-to-sequence model," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020, pp. 708–718.

[42] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of IR techniques," *ACM Trans. Inf. Syst.*, vol. 20, no. 4, pp. 422–446, 2002.

[43] I. Soboroff, S. Huang, and D. Harman, "Trec 2019 news track overview." in *TREC*, 2019.

[44] L. H. Bonifacio, I. Campiotti, R. Lotufo, and R. Nogueira, "mmarco: A multilingual version of ms marco passage ranking dataset," *arXiv preprint arXiv:2108.13897*, 2021.

[45] K. Roberts, T. Alam, S. Bedrick, D. Demner-Fushman, K. Lo, I. Soboroff, E. Voorhees, L. L. Wang, and W. R. Hersh, "Trec-covid: rationale and structure of an information retrieval shared task for covid-19," *Journal of the American Medical Informatics Association*, vol. 27, no. 9, pp. 1431–1436, 2020.

[46] V. Boteva, D. Gholipour, A. Sokolov, and S. Riezler, "A full-text learning to rank dataset for medical information retrieval," in *European Conference on Information Retrieval*. Springer, 2016, pp. 716–722.

[47] A. Suarez, D. Albakour, D. Corney, M. Martinez, and J. Esquivel, "A data collection for evaluating the retrieval of related tweets to news articles," in *European Conference on Information Retrieval*. Springer, 2018, pp. 780–786.

[48] A. Bondarenko, M. Fröbe, M. Beloucif, L. Gienapp, Y. Ajjour, A. Panchenko, C. Biemann, B. Stein, H. Wachsmuth, M. Potthast, and M. Hagen, "Overview of touché 2020: Argument retrieval," in *Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, Thessaloniki, Greece, September 22-25, 2020*, ser. CEUR Workshop Proceedings, vol. 2696, 2020.

[49] H. Wachsmuth, S. Syed, and B. Stein, "Retrieval of the best counterargument without prior topic knowledge," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 241–251.

[50] F. Hasibi, F. Nikolaev, C. Xiong, K. Balog, S. E. Bratsberg, A. Kotov, and J. Callan, "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, Shinjuku, Tokyo, Japan, August 7-11, 2017*, 2017, pp. 1265–1268.

[51] N. Craswell, D. Campos, B. Mitra, E. Yilmaz, and B. Billerbeck, "ORCAS: 20 million clicked query-document pairs for analyzing search," in *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*, 2020, pp. 2983–2989.

[52] C. Sciavolino, Z. Zhong, J. Lee, and D. Chen, "Simple entity-centric questions challenge dense retrievers," in *EMNLP*, 2021.

[53] N. Craswell, B. Mitra, E. Yilmaz, D. Campos, and J. Lin, "Trec 2021 deep learning track guidelines," https://microsoft.github.io/msmarco/TREC-Deep-Learning.html.

[54] Y. Qiu, H. Li, Y. Qu, Y. Chen, Q. She, J. Liu, H. Wu, and H. Wang, "Dureader_retrieval: A large-scale chinese benchmark for passage retrieval from web search engine," *arXiv preprint arXiv:2203.10232*, 2022.

[55] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ questions for machine comprehension of text," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 2383–2392.

[56] M. Joshi, E. Choi, D. Weld, and L. Zettlemoyer, "TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 1601–1611.

[57] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. Cohen, R. Salakhutdinov, and C. D. Manning, "HotpotQA: A dataset for diverse, explainable multi-hop question answering," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 2369–2380.

[58] J. Berant, A. Chou, R. Frostig, and P. Liang, "Semantic parsing on Freebase from question-answer pairs," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013, pp. 1533–1544.

[59] P. Baudiš and J. Šedivỳ, "Modeling of the question answering task in the yodaqa system," in *International Conference of the cross-language evaluation Forum for European languages*. Springer, 2015, pp. 222–228.

[60] M. Maia, S. Handschuh, A. Freitas, B. Davis, R. McDermott, M. Zarrouk, and A. Balahur, "Www'18 open challenge: financial opinion mining and question answering," in *Companion Proceedings of the The Web Conference 2018*, 2018, pp. 1941–1942.

[61] G. Tsatsaronis, G. Balikas, P. Malakasiotis, I. Partalas, M. Zschunke, M. R. Alvers, D. Weissenborn, A. Krithara, S. Petridis, D. Polychronopoulos *et al.*, "An overview of the bioasq large-scale biomedical semantic indexing and question answering competition," *BMC bioinformatics*, vol. 16, no. 1, pp. 1–28, 2015.

[62] D. Hoogeveen, K. M. Verspoor, and T. Baldwin, "Cqadupstack: A benchmark data set for community question-answering research," in *Proceedings of the 20th Australasian document computing symposium*, 2015, pp. 1–8.

[63] S. Iyer, N. Dandekar, and K. Csernai, "First quora dataset release: Question pairs," https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs.

[64] J. Wang, A. Jatowt, and M. Yoshikawa, "Archivalqa: A large-scale benchmark dataset for open domain question answering over archival news collections," *arXiv preprint arXiv:2109.03438*, 2021.

[65] P. Huber, A. Aghajanyan, B. Oğuz, D. Okhonko, W.-t. Yih, S. Gupta, and X. Chen, "Ccqa: A new web-scale question answering dataset for model pre-training," *arXiv preprint arXiv:2110.07731*, 2021.

[66] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal, "FEVER: a large-scale dataset for fact extraction and VERification," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 809–819.

[67] T. Diggelmann, J. Boyd-Graber, J. Bulian, M. Ciaramita, and M. Leippold, "Climate-fever: A dataset for verification of real-world climate claims," *arXiv preprint arXiv:2012.00614*, 2020.

[68] D. Wadden, S. Lin, K. Lo, L. L. Wang, M. van Zuylen, A. Cohan, and H. Hajishirzi, "Fact or fiction: Verifying scientific claims," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 7534–7550.

[69] A. Cohan, S. Feldman, I. Beltagy, D. Downey, and D. Weld, "SPECTER: Document-level representation learning using citation-informed transformers," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 2270–2282.

[70] N. Craswell, B. Mitra, E. Yilmaz, D. F. Campos, and J. J. Lin, "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*, 2021.

[71] N. Arabzadeh, B. Mitra, and E. Bagheri, "Ms marco chameleons: Challenging the ms marco leaderboard with extremely obstinate queries," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 4426–4435.

[72] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, and I. Gurevych, "Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models," *arXiv preprint arXiv:2104.08663*, 2021.

[73] F. Petroni, A. Piktus, A. Fan, P. Lewis, M. Yazdani, N. De Cao, J. Thorne, Y. Jernite, V. Karpukhin, J. Maillard, V. Plachouras, T. Rocktäschel, and S. Riedel, "KILT: a benchmark for knowledge intensive language tasks," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021, pp. 2523–2544.

[74] N. Arabzadeh, A. Vtyurina, X. Yan, and C. L. Clarke, "Shallow pooling for sparse labels," *arXiv preprint arXiv:2109.00062*, 2021.

[75] D. Harman, "Information retrieval evaluation," *Synthesis Lectures on Information Concepts, Retrieval, and Services*, vol. 3, no. 2, pp. 1–119, 2011.

[76] S. Vargas, "Novelty and diversity enhancement and evaluation in recommender systems and information retrieval," in *The 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, Gold Coast , QLD, Australia - July 06 - 11, 2014*, 2014, p. 1281.

[77] M. Zhu, "Recall, precision and average precision," *Department of Statistics and Actuarial Science, University of Waterloo, Waterloo*, vol. 2, no. 30, p. 6, 2004.

[78] L. Gao and J. Callan, "Unsupervised corpus aware language model pre-training for dense passage retrieval," *arXiv preprint arXiv:2108.05540*, 2021.

[79] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*, 2008.

[80] M. Sanderson *et al.*, "Test collection based evaluation of information retrieval systems," *Foundations and Trends® in Information Retrieval*, vol. 4, no. 4, pp. 247–375, 2010.

[81] L. Gao, X. Ma, J. J. Lin, and J. Callan, "Tevatron: An efficient and flexible toolkit for dense retrieval," *ArXiv*, vol. abs/2203.05765, 2022.

[82] J. Lin, X. Ma, S.-C. Lin, J.-H. Yang, R. Pradeep, and R. Nogueira, "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 2021)*, 2021, pp. 2356–2362.

[83] P. Yang, H. Fang, and J. Lin, "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, Shinjuku, Tokyo, Japan, August 7-11, 2017*, 2017, pp. 1253–1256.

[84] J. Johnson, M. Douze, and H. JÃ©gou, "Billion-scale similarity search with gpus," *IEEE Transactions on Big Data*, 2019.

[85] S. Zhuang and G. Zuccon, "Asyncval: A toolkit for asynchronously validating dense retriever checkpoints during training," *ArXiv*, vol. abs/2202.12510, 2022.

[86] Z. Liu, K. Zhang, C. Xiong, Z. Liu, and M. Sun, "Openmatch: An open source library for neu-ir research," in *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, 2021, pp. 2531–2535.

[87] J. Guo, Y. Fan, X. Ji, and X. Cheng, "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 2019, Paris, France, July 21-25, 2019*, 2019, pp. 1297–1300.

[88] C. Macdonald, N. Tonellotto, S. MacAvaney, and I. Ounis, "Pyterrier: Declarative experimentation in python from BM25 to dense retrieval," in *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, 2021, pp. 4526–4533.

[89] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 3982–3992.

[90] K. Wang, N. Reimers, and I. Gurevych, "Tsdae: Using transformer-based sequential denoising auto-encoder for unsupervised sentence embedding learning," in *EMNLP*, 2021.

[91] S. Humeau, K. Shuster, M. Lachaux, and J. Weston, "Poly-encoders: Architectures and pre-training strategies for fast and accurate multi-sentence scoring," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

[92] K. Lee, M.-W. Chang, and K. Toutanova, "Latent retrieval for weakly supervised open domain question answering," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 6086–6096.

[93] W. Chang, F. X. Yu, Y. Chang, Y. Yang, and S. Kumar, "Pre-training tasks for embedding-based large-scale retrieval," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

[94] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang, "REALM: retrieval-augmented language model pre-training," *CoRR*, vol. abs/2002.08909, 2020.

[95] O. Khattab and M. Zaharia, "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 2020, Virtual Event, China, July 25-30, 2020*, 2020, pp. 39–48.

[96] Y. Luan, J. Eisenstein, K. Toutanova, and M. Collins, "Sparse, dense, and attentional representations for text retrieval," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 329–345, 2021.

[97] J. Zhan, J. Mao, Y. Liu, M. Zhang, and S. Ma, "Repbert: Contextualized text embeddings for first-stage retrieval," *CoRR*, vol. abs/2006.15498, 2020.

[98] L. Xiong, C. Xiong, Y. Li, K. Tang, J. Liu, P. N. Bennett, J. Ahmed, and A. Overwijk, "Approximate nearest neighbor negative contrastive learning for dense text retrieval," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

[99] D. Liang, P. Xu, S. Shakeri, C. N. d. Santos, R. Nallapati, Z. Huang, and B. Xiang, "Embedding-based zero-shot retrieval through query generation," *arXiv preprint arXiv:2009.10270*, 2020.

[100] Y. Yang, N. Jin, K. Lin, M. Guo, and D. Cer, "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 (Volume 2: Short Papers)*, 2021, pp. 263–268.

[101] S. Hofstätter, S. Althammer, M. Schröder, M. Sertkan, and A. Hanbury, "Improving efficient neural ranking models with cross-architecture knowledge distillation," *arXiv preprint arXiv:2010.02666*, 2020.

[102] N. Thakur, N. Reimers, J. Daxenberger, and I. Gurevych, "Augmented SBERT: Data augmentation method for improving bi-encoders for pairwise sentence scoring tasks," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021, pp. 296–310.

[103] S.-C. Lin, J.-H. Yang, and J. Lin, "Distilling dense representations for ranking using tightly-coupled teachers," *arXiv preprint arXiv:2010.11386*, 2020.

[104] J. Lu, G. H. Ábrego, J. Ma, J. Ni, and Y. Yang, "Multi-stage training with improved negative contrast for neural passage retrieval," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 6091–6103.

[105] G. Izacard and E. Grave, "Distilling knowledge from reader to retriever for question answering," in *International Conference on Learning Representations*, 2021.

[106] J. Lee, M. Sung, J. Kang, and D. Chen, "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 (Volume 1: Long Papers)*, 2021, pp. 6634–6647.

[107] B. Oguz, X. Chen, V. Karpukhin, S. Peshterliev, D. Okhonko, M. Schlichtkrull, S. Gupta, Y. Mehdad, and S. Yih, "Unik-qa: Unified representations of structured and unstructured knowledge for open-domain question answering," *arXiv preprint arXiv:2012.14610*, 2020.

[108] D. Sachan, M. Patwary, M. Shoeybi, N. Kant, W. Ping, W. L. Hamilton, and B. Catanzaro, "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 (Volume 1: Long Papers)*, 2021, pp. 6648–6662.

[109] L. Gao, Y. Zhang, J. Han, and J. Callan, "Scaling deep contrastive learning batch size under memory limited setup," in *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, 2021, pp. 316–321.

[110] S. Hofstätter, S.-C. Lin, J.-H. Yang, J. Lin, and A. Hanbury, "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*, 2021, pp. 113–122.

[111] J. Zhan, J. Mao, Y. Liu, J. Guo, M. Zhang, and S. Ma, "Optimizing dense retrieval model training with hard negatives," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 1503–1512.

[112] L. Gao and J. Callan, "Condenser: a pre-training architecture for dense retrieval," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 981–993.

[113] H. Tang, X. Sun, B. Jin, J. Wang, F. Zhang, and W. Wu, "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 (Volume 1: Long Papers)*, 2021, pp. 5054–5064.

[114] P. Prakash, J. Killingback, and H. Zamani, "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*, 2021, pp. 1728–1732.

[115] Y. Li, Z. Liu, C. Xiong, and Z. Liu, "More robust dense retrieval with contrastive dual learning," in *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*, 2021, pp. 287–296.

[116] B. Oğuz, K. Lakhotia, A. Gupta, P. Lewis, V. Karpukhin, A. Piktus, X. Chen, S. Riedel, W.-t. Yih, S. Gupta *et al.*, "Domain-matched pre-training tasks for dense retrieval," *arXiv preprint arXiv:2107.13602*, 2021.

[117] J. Zhan, J. Mao, Y. Liu, J. Guo, M. Zhang, and S. Ma, "Jointly optimizing query encoder and product quantization to improve retrieval performance," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 2487–2496.

[118] J. Ni, G. H. 'Abrego, N. Constant, J. Ma, K. B. Hall, D. M. Cer, and Y. Yang, "Sentence-t5: Scalable sentence encoders from pretrained text-to-text models," *ArXiv*, vol. abs/2108.08877, 2021.

[119] H. Yu, C. Xiong, and J. Callan, "Improving query representations for dense retrieval with pseudo relevance feedback," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 3592–3596.

[120] F. Liu, S. Havrylov, Y. Jiao, J. Massiah, and E. Yilmaz, "Transencoder: Unsupervised sentence-pair modelling through self-and mutual-distillations," *arXiv preprint arXiv:2109.13059*, 2021.

[121] H. Zhang, Y. Gong, Y. Shen, J. Lv, N. Duan, and W. Chen, "Adversarial retriever-ranker for dense text retrieval," *arXiv preprint arXiv:2110.03611*, 2021.

[122] J. Zhan, J. Mao, Y. Liu, J. Guo, M. Zhang, and S. Ma, "Learning discrete representations via constrained clustering for effective and efficient dense retrieval," *arXiv preprint arXiv:2110.05789*, 2021.

[123] X. Chen, K. Lakhotia, B. Oğuz, A. Gupta, P. Lewis, S. Peshterliev, Y. Mehdad, S. Gupta, and W.-t. Yih, "Salient phrase aware dense retrieval: Can a dense retriever imitate a sparse one?" *arXiv preprint arXiv:2110.06918*, 2021.

[124] S. Lu, D. He, C. Xiong, G. Ke, W. Malik, Z. Dou, P. Bennett, T. Liu, and A. Overwijk, "Less is more: Pretrain a strong siamese encoder for dense text retrieval using a weak decoder," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, 2021, pp. 2780–2791.

[125] K. Santhanam, O. Khattab, J. Saad-Falcon, C. Potts, and M. Zaharia, "Colbertv2: Effective and efficient retrieval via lightweight late interaction," *arXiv preprint arXiv:2112.01488*, 2021.

[126] K. Wang, N. Thakur, N. Reimers, and I. Gurevych, "Gpl: Generative pseudo labeling for unsupervised domain adaptation of dense retrieval," *arXiv preprint arXiv:2112.07577*, 2021.

[127] O. Ram, G. Shachaf, O. Levy, J. Berant, and A. Globerson, "Learning to retrieve passages without supervision," *ArXiv*, vol. abs/2112.07708, 2021.

[128] P. Lewis, B. Oğuz, W. Xiong, F. Petroni, W. tau Yih, and S. Riedel, "Boosted dense retriever," *ArXiv*, vol. abs/2112.07771, 2021.

[129] J. Ni, C. Qu, J. Lu, Z. Dai, G. H. 'Abrego, J. Ma, V. Zhao, Y. Luan, K. Hall, M.-W. Chang, and Y. Yang, "Large dual encoders are generalizable retrievers," *ArXiv*, vol. abs/2112.07899, 2021.

[130] G. Izacard, M. Caron, L. Hosseini, S. Riedel, P. Bojanowski, A. Joulin, and E. Grave, "Towards unsupervised dense information retrieval with contrastive learning," *ArXiv*, vol. abs/2112.09118, 2021.

[131] J. Zhang, Z. Liu, W. Han, S. Xiao, R. Zheng, Y. Shao, H. Sun, H. Zhu, P. Srinivasan, D. Deng, Q. Zhang, and X. Xie, "Uni-retriever: Towards learning the unified embedding based re-

triever in bing sponsored search," *ArXiv*, vol. abs/2202.06212, 2022.

[132] C. Xu, D. Guo, N. Duan, and J. McAuley, "Laprador: Unsupervised pretrained dense retriever for zero-shot text retrieval," *arXiv preprint arXiv:2203.06169*, 2022.

[133] J. Zhou, X. Li, L. Shang, L. Luo, K. Zhan, E. Hu, X. Zhang, H. Jiang, Z. Cao, F. Yu *et al.*, "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 (Volume 1: Long Papers)*, 2022, pp. 7135–7146.

[134] S. Jeong, J. Baek, S. Cho, S. J. Hwang, and J. C. Park, "Augmenting document representations for dense retrieval with interpolation and perturbation," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2022, pp. 442–452.

[135] S. Zhang, Y. Liang, M. Gong, D. Jiang, and N. Duan, "Multi-view document representation learning for open-domain dense retrieval," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2022, pp. 5990–6000.

[136] S. Hofstätter, O. Khattab, S. Althammer, M. Sertkan, and A. Hanbury, "Introducing neural bag of whole-words with colberter: Contextualized late interactions using enhanced reduction," *ArXiv*, vol. abs/2203.13088, 2022.

[137] S. Zhuang and G. Zuccon, "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*, 2022, pp. 1444–1454.

[138] J. Liu, J. Liu, Y. Yang, J. Wang, W. Wu, D. Zhao, and R. Yan, "Gnn-encoder: Learning a dual-encoder architecture via graph neural networks for passage retrieval," *arXiv preprint arXiv:2204.08241*, 2022.

[139] X. Ma, J. Guo, R. Zhang, Y. Fan, and X. Cheng, "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*, 2022, pp. 848–858.

[140] H. Zeng, H. Zamani, and V. Vinay, "Curriculum learning for dense retrieval distillation," in *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, 2022, pp. 1979–1983.

[141] Y. Lu, Y. Liu, J. Liu, Y. Shi, Z. Huang, S. Feng, Y. Sun, H. Tian, H. Wu, S. Wang, D. Yin, and H. Wang, "Ernie-search: Bridging cross-encoder with dual-encoder via self on-the-fly distillation for dense passage retrieval," *CoRR*, vol. abs/2205.09153, 2022.

[142] Z. Liu and Y. Shao, "Retromae: Pre-training retrieval-oriented transformers via masked auto-encoder," *CoRR*, vol. abs/2205.12035, 2022.

[143] W. Hong, Z. Zhang, J. Wang, and H. Zhao, "Sentence-aware contrastive learning for open-domain passage retrieval," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2022, pp. 1062–1074.

[144] D. S. Sachan, M. Lewis, D. Yogatama, L. Zettlemoyer, J. Pineau, and M. Zaheer, "Questions are all you need to train a dense passage retriever," *arXiv preprint arXiv:2206.10658*, 2022.

[145] L. Wang, N. Yang, X. Huang, B. Jiao, L. Yang, D. Jiang, R. Majumder, and F. Wei, "Simlm: Pre-training with representation bottleneck for dense passage retrieval," *CoRR*, vol. abs/2207.02578, 2022.

[146] X. Chen, J. Luo, B. He, L. Sun, and Y. Sun, "Towards robust dense retrieval via local ranking alignment," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, 2022, pp. 1980–1986.

[147] S.-C. Lin, M. Li, and J. Lin, "Aggretriever: A simple approach to aggregate textual representation for robust dense passage retrieval," *ArXiv*, vol. abs/2208.00511, 2022.

[148] X. Wu, G. Ma, M. Lin, Z. Lin, Z. Wang, and S. Hu, "Contextual mask auto-encoder for dense passage retrieval," *arXiv preprint arXiv:2208.07670*, 2022.

[149] X. Ma, R. Zhang, J. Guo, Y. Fan, and X. Cheng, "A contrastive pre-training approach to learn discriminative autoencoder for dense retrieval," *arXiv preprint arXiv:2208.09846*, 2022.

[150] Z. Tang, B. Wang, and T. Yao, "Dptdr: Deep prompt tuning for dense passage retrieval," *arXiv preprint arXiv:2208.11503*, 2022.

[151] K. Zhang, C. Tao, T. Shen, C. Xu, X. Geng, B. Jiao, and D. Jiang, "Led: Lexicon-enlightened dense retriever for large-scale retrieval," *ArXiv*, vol. abs/2208.13661, 2022.

[152] T. Shen, X. Geng, C. Tao, C. Xu, X. Huang, B. Jiao, L. Yang, and D. Jiang, "Lexmae: Lexicon-bottlenecked pretraining for large-scale retrieval," *CoRR*, vol. abs/2208.14754, 2022.

[153] Z. Dai, V. Zhao, J. Ma, Y. Luan, J. Ni, J. Lu, A. Bakalov, K. Guu, K. B. Hall, and M.-W. Chang, "Promptagator: Few-shot dense retrieval from 8 examples," *ArXiv*, vol. abs/2209.11755, 2022.

[154] Z. Lin, Y. Gong, X. Liu, H. Zhang, C. Lin, A. Dong, J. Jiao, J. Lu, D. Jiang, R. Majumder, and N. Duan, "Prod: Progressive distillation for dense retrieval," 2022.

[155] H. Cheng, H. Fang, X. Liu, and J. Gao, "Task-aware specialization for efficient and robust dense retrieval for open-domain question answering," *arXiv preprint arXiv:2210.05156*, 2022.

[156] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[157] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder–decoder approaches," in *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, 2014, pp. 103–111.

[158] T. Lin, Y. Wang, X. Liu, and X. Qiu, "A survey of transformers," *CoRR*, vol. abs/2106.04554, 2021.

[159] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient transformers: A survey," *CoRR*, vol. abs/2009.06732, 2020.

[160] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[161] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[162] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *ACM Computing Surveys (CSUR)*, 2022.

[163] X. Ma, Z. Wang, P. Ng, R. Nallapati, and B. Xiang, "Universal text representation from bert: An empirical study," *arXiv preprint arXiv:1910.07973*, 2019.

[164] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[165] C. Raffel, N. M. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *ArXiv*, vol. abs/1910.10683, 2020.

[166] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, E. Brynjolfsson, S. Buch, D. Card, R. Castellon, N. S. Chatterji, A. S. Chen, K. Creel, J. Q. Davis, D. Demszky, C. Donahue, M. Doumbouya, E. Durmus, S. Ermon, J. Etchemendy, K. Ethayarajh, L. Fei-Fei, C. Finn, T. Gale, L. Gillespie, K. Goel, N. D. Goodman, S. Grossman, N. Guha, T. Hashimoto, P. Henderson, J. Hewitt, D. E. Ho, J. Hong, K. Hsu, J. Huang, T. Icard, S. Jain, D. Jurafsky, P. Kalluri, S. Karamcheti, G. Keeling, F. Khani, O. Khattab, P. W. Koh, M. S. Krass, R. Krishna, R. Kuditipudi, and et al., "On the opportunities and risks of foundation models," *CoRR*, vol. abs/2108.07258, 2021.

[167] Z. Dai and J. Callan, "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 2020, Virtual Event, China, July 25-30, 2020*, 2020, pp. 1533–1536.

[168] ——, "Context-aware document term weighting for ad-hoc search," in *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, 2020, pp. 1897–1907.

[169] L. Gao, Z. Dai, and J. Callan, "COIL: Revisit exact lexical match in information retrieval with contextualized inverted list," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021, pp. 3030–3042.

[170] J. J. Lin and X. Ma, "A few brief notes on deepimpact, coil, and a conceptual framework for information retrieval techniques," *ArXiv*, vol. abs/2106.14807, 2021.

[171] R. Nogueira, J. Lin, and A. Epistemic, "From doc2query to doctttttquery," *Online preprint*, 2019.

[172] T. Formal, B. Piwowarski, and S. Clinchant, "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*, 2021, pp. 2288–2292.

[173] T. Formal, C. Lassance, B. Piwowarski, and S. Clinchant, "Splade v2: Sparse lexical and expansion model for information retrieval," *arXiv preprint arXiv:2109.10086*, 2021.

[174] Y. Qiao, C. Xiong, Z. Liu, and Z. Liu, "Understanding the behaviors of bert in ranking," *ArXiv*, vol. abs/1904.07531, 2019.

[175] C. Xiong, Z. Dai, J. Callan, Z. Liu, and R. Power, "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, Shinjuku, Tokyo, Japan, August 7-11, 2017*, 2017, pp. 55–64.

[176] Y. Sun, S. Wang, Y. Li, S. Feng, H. Tian, H. Wu, and H. Wang, "ERNIE 2.0: A continual pre-training framework for language understanding," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, 2020, pp. 8968–8975.

[177] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 3982–3992.

[178] A. Conneau and D. Kiela, "SentEval: An evaluation toolkit for universal sentence representations," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[179] W. Kong, S. Khadanga, C. Li, S. K. Gupta, M. Zhang, W. Xu, and M. Bendersky, "Multi-aspect dense retrieval," in *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, 2022, pp. 3178–3186.

[180] M. Seo, J. Lee, T. Kwiatkowski, A. Parikh, A. Farhadi, and H. Hajishirzi, "Real-time open-domain question answering with dense-sparse phrase index," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 4430–4441.

[181] J. Lee, A. Wettig, and D. Chen, "Phrase retrieval learns passage retrieval, too," *arXiv preprint arXiv:2109.08133*, 2021.

[182] M. Seo, T. Kwiatkowski, A. Parikh, A. Farhadi, and H. Hajishirzi, "Phrase-indexed question answering: A new challenge for scalable document comprehension," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 559–564.

[183] Y. Liu, K. Hashimoto, Y. Zhou, S. Yavuz, C. Xiong, and S. Y. Philip, "Dense hierarchical retrieval for open-domain question answering," in *Findings of the Association for Computational Linguistics: EMNLP 2021*, 2021, pp. 188–200.

[184] M. Li, M. Li, K. Xiong, and J. Lin, "Multi-task dense retrieval via model uncertainty fusion for open-domain question answering," in *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, 2021, pp. 274–287.

[185] N. Tonellotto and C. Macdonald, "Query embedding pruning for dense retrieval," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 3453–3457.

[186] H. Li, A. Mourad, S. Zhuang, B. Koopman, and G. Zuccon, "Pseudo relevance feedback with deep language models and dense retrievers: Successes and pitfalls," *arXiv preprint arXiv:2108.11044*, 2021.

[187] X. Wang, C. Macdonald, N. Tonellotto, and I. Ounis, "Pseudo-relevance feedback for multiple representation dense retrieval," *arXiv preprint arXiv:2106.11251*, 2021.

[188] H. Li, S. Zhuang, A. Mourad, X. Ma, J. J. Lin, and G. Zuccon, "Improving query representations for dense retrieval with pseudo relevance feedback: A reproducibility study," *ArXiv*, vol. abs/2112.06400, 2021.

[189] S. Zhuang, H. Li, and G. Zuccon, "Implicit feedback for dense passage retrieval: A counterfactual approach," *ArXiv*, vol. abs/2204.00718, 2022.

[190] X. Wang, C. Macdonald, N. Tonellotto, and I. Ounis, "Pseudo-relevance feedback for multiple representation dense retrieval," in *ICTIR '21: The 2021 ACM SIGIR International Conference on the Theory of Information Retrieval, Virtual Event, Canada, July 11, 2021*, 2021, pp. 297–306.

[191] W. L. Tam, X. Liu, K. Ji, L. Xue, X. Zhang, Y. Dong, J. Liu, M. Hu, and J. Tang, "Parameter-efficient prompt tuning makes generalized and calibrated neural text retrievers," *ArXiv*, vol. abs/2207.07087, 2022.

[192] X. Liu, K. Ji, Y. Fu, Z. Du, Z. Yang, and J. Tang, "P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks," *ArXiv*, vol. abs/2110.07602, 2021.

[193] R. Nogueira and K. Cho, "Passage re-ranking with BERT," *CoRR*, vol. abs/1901.04085, 2019.

[194] L. Liu, P. Lewis, S. Riedel, and P. Stenetorp, "Challenges in generalization in open domain question answering," *ArXiv*, vol. abs/2109.01156, 2021.

[195] N. Reimers and I. Gurevych, "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 (Volume 2: Short Papers)*, 2021, pp. 605–611.

[196] H. Fang, T. Tao, and C. Zhai, "A formal study of information retrieval heuristics," in *SIGIR '04*, 2004.

[197] H. Fang and C. Zhai, "An exploration of axiomatic approaches to information retrieval," in *SIGIR '05*, 2005.

[198] ——, "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*, 2006.

[199] C. Rosset, B. Mitra, C. Xiong, N. Craswell, X. Song, and S. Tiwary, "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 2019, Paris, France, July 21-25, 2019*, B. Piwowarski, M. Chevalier, É. Gaussier, Y. Maarek, J. Nie, and F. Scholer, Eds. ACM, 2019, pp. 981–984.

[200] M. Völske, A. Bondarenko, M. Fröbe, B. Stein, J. Singh, M. Hagen, and A. Anand, "Towards axiomatic explanations for neural ranking models," in *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*, 2021, pp. 13–22.

[201] A. Câmara and C. Hauff, "Diagnosing BERT with retrieval heuristics," in *Advances in Information Retrieval - 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14-17, 2020, Proceedings, Part I*, vol. 12035, 2020, pp. 605–618.

[202] S. MacAvaney, S. Feldman, N. Goharian, D. Downey, and A. Cohan, "Abnirml: Analyzing the behavior of neural ir models," *arXiv preprint arXiv:2011.00696*, 2020.

[203] T. Formal, B. Piwowarski, and S. Clinchant, "A white box analysis of colbert," in *ECIR*, 2021.

[204] J. Lu, G. H. Ábrego, J. Ma, J. Ni, and Y. Yang, "Neural passage retrieval with improved negative contrast," *CoRR*, vol. abs/2010.12523, 2020.

[205] N. Arabzadeh, X. Yan, and C. L. Clarke, "Predicting efficiency/effectiveness trade-offs for dense vs. sparse retrieval strategy selection," *arXiv preprint arXiv:2109.10739*, 2021.

[206] S.-C. Lin and J. J. Lin, "Densifying sparse representations for passage retrieval by representational slicing," *ArXiv*, vol. abs/2112.04666, 2021.

[207] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *CoRR*, vol. abs/1807.03748, 2018.

[208] B. Dodla, A. K. Mohankumar, and A. Singh, "HEARTS: multi-task fusion of dense retrieval and non-autoregressive generation for sponsored search," *CoRR*, vol. abs/2209.05861, 2022.

[209] N. Yang, F. Wei, B. Jiao, D. Jiang, and L. Yang, "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 (Volume 1: Long Papers)*, 2021, pp. 6120–6129.

[210] Z. Li, N. Yang, L. Wang, and F. Wei, "Learning diverse document representations with deep query interactions for dense retrieval," *arXiv preprint arXiv:2208.04232*, 2022.

[211] X. Ma, J. Guo, R. Zhang, Y. Fan, X. Ji, and X. Cheng, "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*, 2021, pp. 283–291.

[212] M. Henderson, R. Al-Rfou, B. Strope, Y.-H. Sung, L. Lukács, R. Guo, S. Kumar, B. Miklos, and R. Kurzweil, "Efficient natural language response suggestion for smart reply," *arXiv preprint arXiv:1705.00652*, 2017.

[213] M. E. Maron and J. L. Kuhns, "On relevance, probabilistic indexing and information retrieval," *Journal of the ACM (JACM)*, vol. 7, no. 3, pp. 216–244, 1960.

[214] Y. Cai, J. Guo, Y. Fan, Q. Ai, R. Zhang, and X. Cheng, "Hard negatives or false negatives: Correcting pooling bias in training neural ranking models," *arXiv preprint arXiv:2209.05072*, 2022.

[215] K. Zhou, B. Zhang, X. Zhao, and J. Wen, "Debiased contrastive learning of unsupervised sentence representations," in *ACL*, 2022, pp. 6120–6130.

[216] K. Zhou, Y. Gong, X. Liu, W. X. Zhao, Y. Shen, A. Dong, J. Lu, R. Majumder, J.-R. Wen, N. Duan, and W. Chen, "Simans: Simple ambiguous negatives sampling for dense text retrieval," in *EMNLP*, 2022.

[217] Y. Goldberg and O. Levy, "word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method," *arXiv preprint arXiv:1402.3722*, 2014.

[218] Z. Yang, M. Ding, C. Zhou, H. Yang, J. Zhou, and J. Tang, "Understanding negative sampling in graph representation learning," in *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, 2020, pp. 1666–1676.

[219] J. Maillard, V. Karpukhin, F. Petroni, W.-t. Yih, B. Oğuz, V. Stoyanov, and G. Ghosh, "Multi-task retrieval for knowledge-intensive tasks," *arXiv preprint arXiv:2101.00117*, 2021.

[220] R. Ren, Y. Qu, J. Liu, W. X. Zhao, Q. Wu, Y. Ding, H. Wu, H. Wang, and J. Wen, "A thorough examination on zero-shot dense retrieval," *CoRR*, vol. abs/2204.12755, 2022.

[221] J. Zhan, X. Xie, J. Mao, Y. Liu, M. Zhang, and S. Ma, "Evaluating extrapolation performance of dense retrieval," *CoRR*, vol. abs/2204.11447, 2022.

[222] X. Shen, S. Vakulenko, M. D. Tredici, G. Barlacchi, B. Byrne, and A. de Gispert, "Low-resource dense retrieval for open-domain question answering: A comprehensive survey," *CoRR*, vol. abs/2208.03197, 2022.

[223] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *CoRR*, vol. abs/1503.02531, 2015.

[224] S. Yu, Z. Liu, C. Xiong, T. Feng, and Z. Liu, "Few-shot conversational dense retrieval," in *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, 2021, pp. 829–838.

[225] M. Rezagholizadeh, A. Jafari, P. S. M. Saladi, P. Sharma, A. S. Pasand, and A. Ghodsi, "Pro-kd: Progressive distillation by following the footsteps of the teacher," in *Proceedings of the 29th International Conference on Computational Linguistics, COLING 2022, Gyeongju, Republic of Korea, October 12-17, 2022*, 2022, pp. 4714–4727.

[226] C. Zhou, J. Gu, and G. Neubig, "Understanding knowledge distillation in non-autoregressive machine translation," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

[227] C. Alberti, D. Andor, E. Pitler, J. Devlin, and M. Collins, "Synthetic QA corpora generation with roundtrip consistency," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 6168–6173.

[228] P. Lewis, Y. Wu, L. Liu, P. Minervini, H. Küttler, A. Piktus, P. Stenetorp, and S. Riedel, "Paq: 65 million probably-asked questions and what you can do with them," *arXiv preprint arXiv:2102.07033*, 2021.

[229] T. Gao, X. Yao, and D. Chen, "Simcse: Simple contrastive learning of sentence embeddings," in *EMNLP*, 2021.

[230] Y. Yan, R. Li, S. Wang, F. Zhang, W. Wu, and W. Xu, "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 (Volume 1: Long Papers)*, 2021, pp. 5065–5075.

[231] X. Ma, J. Guo, R. Zhang, Y. Fan, Y. Li, and X. Cheng, "B-PROP: bootstrapped pre-training with representative words prediction for ad-hoc retrieval," in *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, 2021, pp. 1318–1327.

[232] S. Shakeri, C. Nogueira dos Santos, H. Zhu, P. Ng, F. Nan, Z. Wang, R. Nallapati, and B. Xiang, "End-to-end synthetic data generation for domain adaptation of question answering systems," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 5445–5460.

[233] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "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*, 2020, pp. 7871–7880.

[234] R. G. Reddy, V. Yadav, M. A. Sultan, M. Franz, V. Castelli, H. Ji, and A. Sil, "Towards robust neural retrieval models with synthetic pre-training," *arXiv preprint arXiv:2104.07800*, 2021.

[235] J. Ma, I. Korotkov, Y. Yang, K. Hall, and R. McDonald, "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: Main Volume*, 2021, pp. 1075–1088.

[236] V. Balachandran, A. Vaswani, Y. Tsvetkov, and N. Parmar, "Simple and efficient ways to improve realm," *arXiv preprint arXiv:2104.08710*, 2021.

[237] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, "Semi-supervised recursive autoencoders for predicting sentiment distributions," in *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, 2011, pp. 151–161.

[238] R. Socher, E. H. Huang, J. Pennington, A. Y. Ng, and C. D. Manning, "Dynamic pooling and unfolding recursive autoencoders for paraphrase detection," in *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, 2011, pp. 801–809.

[239] J. Li, T. Luong, and D. Jurafsky, "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 (Volume 1: Long Papers)*, 2015, pp. 1106–1115.

[240] A. M. Dai and Q. V. Le, "Semi-supervised sequence learning," in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, 2015, pp. 3079–3087.

[241] N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," in *2015 IEEE Information Theory Workshop, ITW 2015, Jerusalem, Israel, April 26 - May 1, 2015*, 2015, pp. 1–5.

[242] A. Liu and S. Yang, "Masked autoencoders as the unified learners for pre-trained sentence representation," *CoRR*, vol. abs/2208.00231, 2022.

[243] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, "A simple framework for contrastive learning of visual representations," in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, ser. Proceedings of Machine Learning Research, vol. 119, 2020, pp. 1597–1607.

[244] K. He, H. Fan, Y. Wu, S. Xie, and R. B. Girshick, "Momentum contrast for unsupervised visual representation learning," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, 2020, pp. 9726–9735.

[245] A. Neelakantan, T. Xu, R. Puri, A. Radford, J. M. Han, J. Tworek, Q. Yuan, N. A. Tezak, J. W. Kim, C. Hallacy, J. Heidecke, P. Shyam, B. Power, T. E. Nekoul, G. Sastry, G. Krueger, D. P. Schnurr, F. P. Such, K. S.-K. Hsu, M. Thompson, T. Khan, T. Sherbakov, J. Jang, P. Welinder, and L. Weng, "Text and code embeddings by contrastive pre-training," *ArXiv*, vol. abs/2201.10005, 2022.

[246] Y. Ma, D. Yu, T. Wu, and H. Wang, "Paddlepaddle: An open-source deep learning platform from industrial practice," *Frontiers of Data and Domputing*, vol. 1, no. 1, pp. 105–115, 2019.

[247] K. J. McDonell, "An inverted index implementation," *The Computer Journal*, vol. 20, no. 2, pp. 116–123, 1977.

[248] A. Shrivastava and P. Li, "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 2014, December 8-13 2014, Montreal, Quebec, Canada, 2014, pp. 2321–2329.

[249] N. Bhatia and Vandana, "Survey of nearest neighbor techniques," *CoRR*, vol. abs/1007.0085, 2010.

[250] K. Hajebi, Y. Abbasi-Yadkori, H. Shahbazi, and H. Zhang, "Fast approximate nearest-neighbor search with k-nearest neighbor graph," in *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, 2011, pp. 1312–1317.

[251] W. Li, Y. Zhang, Y. Sun, W. Wang, M. Li, W. Zhang, and X. Lin, "Approximate nearest neighbor search on high dimensional data—experiments, analyses, and improvement," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 8, pp. 1475–1488, 2019.

[252] M. Aumüller, E. Bernhardsson, and A. Faithfull, "Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms," in *SISAP*, 2017.

[253] H. V. Simhadri, G. R. Williams, M. Aumüller, M. Douze, A. Babenko, D. Baranchuk, Q. Chen, L. Hosseini, R. Krishnaswamy, G. Srinivasa, S. J. Subramanya, and J. Wang, "Results of the neurips'21 challenge on billion-scale approximate nearest neighbor search," *ArXiv*, vol. abs/2205.03763, 2022.

[254] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *STOC '98*, 1998.

[255] J. Wang, T. Zhang, J. Song, N. Sebe, and H. T. Shen, "A survey on learning to hash," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, pp. 769–790, 2018.

[256] J. Sivic and A. Zisserman, "Video google: a text retrieval approach to object matching in videos," *Proceedings Ninth IEEE International Conference on Computer Vision*, pp. 1470–1477 vol.2, 2003.

[257] Q. Chen, B. Zhao, H. Wang, M. Li, C. Liu, Z. Li, M. Yang, and J. Wang, "Spann: Highly-efficient billion-scale approximate nearest neighbor search," in *NeurIPS*, 2021.

[258] J. Wang and S. Li, "Query-driven iterated neighborhood graph search for large scale indexing," *Proceedings of the 20th ACM international conference on Multimedia*, 2012.

[259] Y. Malkov, A. Ponomarenko, A. Logvinov, and V. Krylov, "Approximate nearest neighbor algorithm based on navigable small world graphs," *Inf. Syst.*, vol. 45, pp. 61–68, 2014.

[260] J. M. Kleinberg, "The small-world phenomenon: an algorithmic perspective," in *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, 2000, pp. 163–170.

[261] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 117–128, 2011.

[262] T. Ge, K. He, Q. Ke, and J. Sun, "Optimized product quantization for approximate nearest neighbor search," in *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*, 2013, pp. 2946–2953.

[263] H. Zhang, H. Shen, Y. Qiu, Y. Jiang, S. Wang, S. Xu, Y. Xiao, B. Long, and W.-Y. Yang, "Joint learning of deep retrieval model and product quantization based embedding index," *arXiv preprint arXiv:2105.03933*, 2021.

[264] I. Yamada, A. Asai, and H. Hajishirzi, "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 (Volume 2: Short Papers)*, 2021, pp. 979–986.

[265] J. Zhan, J. Mao, Y. Liu, J. Guo, M. Zhang, and S. Ma, "Interpreting dense retrieval as mixture of topics," *arXiv preprint arXiv:2111.13957*, 2021.

[266] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 4, pp. 824–836, 2018.

[267] Q. Chen, H. Wang, M. Li, G. Ren, S. Li, J. Zhu, J. Li, C. Liu, L. Zhang, and J. Wang, *SPTAG: A library for fast approximate nearest neighbor search*, 2018.

[268] R. Guo, P. Sun, E. Lindgren, Q. Geng, D. Simcha, F. Chern, and S. Kumar, "Accelerating large-scale inference with anisotropic vector quantization," in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, ser. Proceedings of Machine Learning Research, vol. 119, 2020, pp. 3887–3896.

[269] A. Piktus, F. Petroni, V. Karpukhin, D. Okhonko, S. Broscheit, G. Izacard, P. Lewis, B. Oguz, E. Grave, W. Yih, and S. Riedel, "The web is your oyster - knowledge-intensive NLP against a very large web corpus," *CoRR*, vol. abs/2112.09924, 2021.

[270] S. J. Subramanya, F. Devvrit, H. V. Simhadri, R. Krishnaswamy, and R. Kadekodi, "Diskann: Fast accurate billion-point nearest neighbor search on a single node," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 2019.

[271] R. Baeza-Yates and B. A. Ribeiro-Neto, *Modern Information Retrieval - the concepts and technology behind search, Second edition*, 2011.

[272] Y. Qiao, C. Xiong, Z. Liu, and Z. Liu, "Understanding the behaviors of BERT in ranking," *CoRR*, vol. abs/1904.07531, 2019.

[273] Z. Wang, P. Ng, X. Ma, R. Nallapati, and B. Xiang, "Multi-passage BERT: A globally normalized BERT model for open-domain question answering," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 5878–5882.

[274] M. Yan, C. Li, C. Wu, B. Bi, W. Wang, J. Xia, and L. Si, "IDST at TREC 2019 deep learning track: Deep cascade ranking with generation-based document expansion and pre-trained language modeling," in *Proceedings of the Twenty-Eighth Text REtrieval Conference, TREC 2019, Gaithersburg, Maryland, USA, November 13-15, 2019*, ser. NIST Special Publication, vol. 1250, 2019.

[275] R. Nogueira, W. Yang, K. Cho, and J. Lin, "Multi-stage document ranking with BERT," *CoRR*, vol. abs/1910.14424, 2019.

[276] Y. Zhou, T. Shen, X. Geng, C. Tao, C. Xu, G. Long, B. Jiao, and D. Jiang, "Towards robust ranker for text retrieval," *ArXiv*, vol. abs/2206.08063, 2022.

[277] L. Gao, Z. Dai, and J. Callan, "Rethink training of BERT rerankers in multi-stage retrieval pipeline," in *Advances in Information Retrieval - 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 - April 1, 2021, Proceedings, Part II*, vol. 12657, 2021, pp. 280–286.

[278] H. Zhuang, Z. Qin, R. Jagerman, K. Hui, J. Ma, J. Lu, J. Ni, X. Wang, and M. Bendersky, "Rankt5: Fine-tuning T5 for text ranking with ranking losses," *CoRR*, vol. abs/2210.10634, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2210.10634

[279] Y. Zhang, D. Long, G. Xu, and P. Xie, "HLATR: enhance multi-stage text retrieval with hybrid list aware transformer reranking," *CoRR*, vol. abs/2205.10569, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2205.10569

[280] L. Pang, J. Xu, Q. Ai, Y. Lan, X. Cheng, and J. Wen, "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*, 2020.

[281] J. Huang, A. Sharma, S. Sun, L. Xia, D. Zhang, P. Pronin, J. Padmanabhan, G. Ottaviano, and L. Yang, "Embedding-based retrieval in facebook search," in *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, 2020, pp. 2553–2561.

[282] A. Magnani, F. Liu, S. Chaidaroon, S. Yadav, P. R. Suram, A. Puthenputhussery, S. Chen, M. Xie, A. Kashi, T. Lee, and C. Liao, "Semantic retrieval at walmart," in *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, 2022, pp. 3495–3503.

[283] S. MacAvaney, A. Yates, A. Cohan, and N. Goharian, "CEDR: contextualized embeddings for document ranking," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, 2019, pp. 1101–1104.

[284] Z. Ma, Z. Dou, W. Xu, X. Zhang, H. Jiang, Z. Cao, and J.-R. Wen, "Pre-training for ad-hoc retrieval: Hyperlink is also you need," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 1212–1221.

[285] Z. Cao, T. Qin, T. Liu, M. Tsai, and H. Li, "Learning to rank: from pairwise approach to listwise approach," in *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, ser. ACM International Conference Proceeding Series, vol. 227, 2007, pp. 129–136.

[286] G. Izacard and E. Grave, "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*, 2021, pp. 874–880.

[287] P. S. H. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[288] J. Wei, M. Bosma, V. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le, "Finetuned language models are zero-shot learners," *ArXiv*, vol. abs/2109.01652, 2022.

[289] T. Chen, M. Zhang, J. Lu, M. Bendersky, and M.-A. Najork, "Out-of-domain semantics to the rescue! zero-shot hybrid retrieval models," in *Advances in Information Retrieval - 44th European Conference on IR Research, ECIR 2022, Stavanger, Norway, April 10-14, 2022, Proceedings, Part I*, 2022.

[290] H. Fun, S. Gandhi, and S. Ravi, "Efficient retrieval optimized multi-task learning," *arXiv preprint arXiv:2104.10129*, 2021.

[291] S. Zhuang and G. Zuccon, "Dealing with typos for bert-based passage retrieval and ranking," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, 2021, pp. 2836–2842.

[292] G. Penha, A. Câmara, and C. Hauff, "Evaluating the robustness of retrieval pipelines with query variation generators," in *Advances in Information Retrieval - 44th European Conference on IR Research, ECIR 2022, Stavanger, Norway, April 10-14, 2022, Proceedings, Part I*, ser. Lecture Notes in Computer Science, vol. 13185, 2022, pp. 397–412.

[293] G. Sidiropoulos and E. Kanoulas, "Analysing the robustness of dual encoders for dense retrieval against misspellings," in *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, 2022, pp. 2132–2136.

[294] C. Carpineto and G. Romano, "A survey of automatic query expansion in information retrieval," *Acm Computing Surveys (CSUR)*, vol. 44, no. 1, pp. 1–50, 2012.

[295] Y. Wang, L. Lyu, and A. Anand, "BERT rankers are brittle: A study using adversarial document perturbations," in *ICTIR '22: The 2022 ACM SIGIR International Conference on the Theory of Information Retrieval, Madrid, Spain, July 11 - 12, 2022*, 2022, pp. 115–120.

[296] J. Liu, Y. Kang, D. Tang, K. Song, C. Sun, X. Wang, W. Lu, and X. Liu, "Order-disorder: Imitation adversarial attacks for black-box neural ranking models," *CoRR*, vol. abs/2209.06506, 2022.

[297] D. Metzler, Y. Tay, D. Bahri, and M. Najork, "Rethinking search: Making experts out of dilettantes," *CoRR*, vol. abs/2105.02274, 2021.

[298] Y. Tay, V. Q. Tran, M. Dehghani, J. Ni, D. Bahri, H. Mehta, Z. Qin, K. Hui, Z. Zhao, J. Gupta, T. Schuster, W. W. Cohen, and D. Metzler, "Transformer memory as a differentiable search index," *ArXiv*, vol. abs/2202.06991, 2022.

[299] N. D. Cao, G. Izacard, S. Riedel, and F. Petroni, "Autoregressive entity retrieval," *ArXiv*, vol. abs/2010.00904, 2021.

[300] H. Lee, S. Yang, H. Oh, and M. Seo, "Generative retrieval for long sequences," *CoRR*, vol. abs/2204.13596, 2022.

[301] Y. Zhou, J. Yao, Z. Dou, L. Y. Wu, and J. rong Wen, "Dynamicretriever: A pre-training model-based ir system with neither sparse nor dense index," *ArXiv*, vol. abs/2203.00537, 2022.

[302] M. Bevilacqua, G. Ottaviano, P. Lewis, W. tau Yih, S. Riedel, and F. Petroni, "Autoregressive search engines: Generating substrings as document identifiers," *ArXiv*, vol. abs/2204.10628, 2022.

[303] Y. Wang, Y. Hou, H. Wang, Z. Miao, S. Wu, H. Sun, Q. Chen, Y. Xia, C. Chi, G. Zhao, Z. Liu, X. Xie, H. A. Sun, W. Deng, Q. Zhang, and M. Yang, "A neural corpus indexer for document retrieval," *CoRR*, vol. abs/2206.02743, 2022.

[304] S. Zhuang, H. Ren, L. Shou, J. Pei, M. Gong, G. Zuccon, and D. Jiang, "Bridging the gap between indexing and retrieval for differentiable search index with query generation," *CoRR*, vol. abs/2206.10128, 2022.

[305] Y. Zhou, J. Yao, Z. Dou, L. Wu, P. Zhang, and J. Wen, "Ultron: An ultimate retriever on corpus with a model-based indexer," *CoRR*, vol. abs/2208.09257, 2022.

[306] J. Chen, R. Zhang, J. Guo, Y. Fan, and X. Cheng, "GERE: generative evidence retrieval for fact verification," in *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, 2022, pp. 2184–2189.

[307] J. Chen, R. Zhang, J. Guo, Y. Liu, Y. Fan, and X. Cheng, "Corpusbrain: Pre-train a generative retrieval model for knowledge-intensive language tasks," *CoRR*, vol. abs/2208.07652, 2022.

[308] Y. Bengio, R. Ducharme, and P. Vincent, "A neural probabilistic language model," in *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA*, 2000, pp. 932–938.

[309] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, 2013, pp. 3111–3119.

[310] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 2227–2237.

[311] A. Radford and K. Narasimhan, "Improving language understanding by generative pre-training," ., 2018.

[312] F. Petroni, T. Rocktäschel, S. Riedel, P. Lewis, A. Bakhtin, Y. Wu, and A. Miller, "Language models as knowledge bases?" in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 2463–2473.

[313] U. Khandelwal, O. Levy, D. Jurafsky, L. Zettlemoyer, and M. Lewis, "Generalization through memorization: Nearest neighbor language models," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

[314] J. He, G. Neubig, and T. Berg-Kirkpatrick, "Efficient nearest neighbor language models," in *EMNLP*, 2021.

[315] D. Yogatama, C. de Masson d'Autume, and L. Kong, "Adaptive semiparametric language models," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 362–373, 2021.

[316] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. van den Driessche, J.-B. Lespiau, B. Damoc, A. Clark, D. de Las Casas, A. Guy, J. Menick, R. Ring, T. W. Hennigan, S. Huang, L. Maggiore, C. Jones, A. Cassirer, A. Brock, M. Paganini, G. Irving, O. Vinyals, S. Osindero, K. Simonyan, J. W. Rae, E. Elsen, and L. Sifre, "Improving language models by retrieving from trillions of tokens," *ArXiv*, vol. abs/2112.04426, 2021.

[317] M. J. Zhang and E. Choi, "Situatedqa: Incorporating extra-linguistic contexts into qa," *arXiv preprint arXiv:2109.06157*, 2021.

[318] J. Herzig, T. Müller, S. Krichene, and J. Eisenschlos, "Open domain question answering over tables via dense retrieval," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021, pp. 512–519.

[319] B. Kostić, J. Risch, and T. Möller, "Multi-modal retrieval of tables and texts using tri-encoder models," in *Proceedings of the 3rd Workshop on Machine Reading for Question Answering*, 2021, pp. 82–91.

[320] X. Zhang, X. Ma, P. Shi, and J. Lin, "Mr. tydi: A multi-lingual benchmark for dense retrieval," *CoRR*, vol. abs/2108.08787, 2021.

[321] A. Asai, J. Kasai, J. Clark, K. Lee, E. Choi, and H. Hajishirzi, "XOR QA: Cross-lingual open-retrieval question answering," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021, pp. 547–564.

[322] A. Asai, X. Yu, J. Kasai, and H. Hajishirzi, "One question answering model for many languages with cross-lingual dense passage retrieval," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[323] T. Pires, E. Schlinger, and D. Garrette, "How multilingual is multilingual BERT?" in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 4996–5001.

[324] W. Zhong, J. Yang, and J. Lin, "Evaluating token-level and passage-level dense retrieval models for math information retrieval," *CoRR*, vol. abs/2203.11163, 2022.

[325] S. Lu, N. Duan, H. Han, D. Guo, S. Hwang, and A. Svyatkovskiy, "Reacc: A retrieval-augmented code completion framework," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, 2022, pp. 6227–6240.

[326] M. Luo, A. Mitra, T. Gokhale, and C. Baral, "Improving biomedical information retrieval with neural retrievers," in *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, 2022, pp. 11 038–11 046.

[327] K. Wang, Q. Yin, W. Wang, S. Wu, and L. Wang, "A comprehensive survey on cross-modal retrieval," *CoRR*, vol. abs/1607.06215, 2016.

[328] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, ser. Proceedings of Machine Learning Research, vol. 139, 2021, pp. 8748–8763.

[329] Y. Jiang, S. Bordia, Z. Zhong, C. Dognin, M. Singh, and M. Bansal, "HoVer: A dataset for many-hop fact extraction and claim verification," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020, pp. 3441–3460.

[330] A. Asai, K. Hashimoto, H. Hajishirzi, R. Socher, and C. Xiong, "Learning to retrieve reasoning paths over wikipedia graph for question answering," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

[331] O. Khattab, C. Potts, and M. A. Zaharia, "Baleen: Robust multi-hop reasoning at scale via condensed retrieval," *ArXiv*, vol. abs/2101.00436, 2021.

[332] W. Xiong, X. L. Li, S. Iyer, J. Du, P. Lewis, W. Y. Wang, Y. Mehdad, W.-t. Yih, S. Riedel, D. Kiela, and B. Oğuz, "Answering complex open-domain questions with multi-hop dense retrieval," *International Conference on Learning Representations*, 2021.

[333] Q. Zhang, S. Chen, D. Xu, Q. Cao, X. Chen, T. Cohn, and M. Fang, "A survey for efficient open domain question answering," *arXiv preprint arXiv:2211.07886*, 2022.

[334] M. Francis-Landau, G. Durrett, and D. Klein, "Capturing semantic similarity for entity linking with convolutional neural networks," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 1256–1261.

[335] N. Gupta, S. Singh, and D. Roth, "Entity linking via joint encoding of types, descriptions, and context," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 2681–2690.

[336] O.-E. Ganea and T. Hofmann, "Deep joint entity disambiguation with local neural attention," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 2619–2629.

[337] L. Logeswaran, M.-W. Chang, K. Lee, K. Toutanova, J. Devlin, and H. Lee, "Zero-shot entity linking by reading entity descriptions," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 3449–3460.

[338] D. Adiwardana, M.-T. Luong, D. R. So, J. Hall, N. Fiedel, R. Thoppilan, Z. Yang, A. Kulshreshtha, G. Nemade, Y. Lu, and Q. V. Le, "Towards a human-like open-domain chatbot," *ArXiv*, vol. abs/2001.09977, 2020.

[339] S. Roller, E. Dinan, N. Goyal, D. Ju, M. Williamson, Y. Liu, J. Xu, M. Ott, E. M. Smith, Y.-L. Boureau, and J. Weston, "Recipes for building an open-domain chatbot," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, 2021, pp. 300–325.

[340] S. Bao, H. He, F. Wang, H. Wu, H. Wang, W. Wu, Z. Guo, Z. Liu, and X. Xu, "PLATO-2: Towards building an open-domain chatbot via curriculum learning," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 2021, pp. 2513–2525.

[341] R. Thoppilan, D. D. Freitas, J. Hall, N. M. Shazeer, A. Kulshreshtha, H.-T. Cheng, A. Jin, T. Bos, L. Baker, Y. Du, Y. Li, H. Lee, H. S. Zheng, A. Ghafouri, M. Menegali, Y. Huang, M. Krikun, D. Lepikhin, J. Qin, D. Chen, Y. Xu, Z. Chen, A. Roberts, M. Bosma, Y. Zhou, C.-C. Chang, I. A. Krivokon, W. J. Rusch, M. Pickett, K. S. Meier-Hellstern, M. R. Morris, T. Doshi, R. D. Santos, T. Duke, J. H. Søraker, B. Zevenbergen, V. Prabhakaran, M. Diaz, B. Hutchinson, K. Olson, A. Molina, E. Hoffman-John, J. Lee, L. Aroyo, R. Rajakumar, A. Butryna, M. Lamm, V. O. Kuzmina, J. Fenton, A. Cohen, R. Bernstein, R. Kurzweil, B. Aguera-Arcas, C. Cui, M. Croak, E. Chi, and Q. H. Le, "Lamda: Language models for dialog applications," *ArXiv*, vol. abs/2201.08239, 2022.

[342] K. Shuster, S. Poff, M. Chen, D. Kiela, and J. Weston, "Retrieval augmentation reduces hallucination in conversation," in *EMNLP*, 2021.

[343] X. Huang, H. He, S. Bao, F. Wang, H. Wu, and H. Wang, "Plato-kag: Unsupervised knowledge-grounded conversation via joint modeling," *Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI*, 2021.

[344] M. Komeili, K. Shuster, and J. Weston, "Internet-augmented dialogue generation," *ArXiv*, vol. abs/2107.07566, 2021.

[345] Y. Liu, G. Huang, J. Liu, W. Lu, S. Cheng, Y. Li, D. Shi, S. Wang, Z. Cheng, and D. Yin, "Pre-trained language model for web-scale retrieval in baidu search," *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021.

[346] M. Fan, J. Guo, S. Zhu, S. Miao, M. Sun, and P. Li, "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 & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, 2019, pp. 2509–2517.

[347] S. Li, F. Lv, T. Jin, G. Lin, K. Yang, X. Zeng, X.-M. Wu, and Q. Ma, "Embedding-based product retrieval in taobao search," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, p. 3181–3189.

[348] Y. Liu, K. Rangadurai, Y. He, S. Malreddy, X. Gui, X. Liu, and F. Borisyuk, "Que2search: Fast and accurate query and document understanding for search at facebook," *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021.

[349] T. Qin, W. Chen, and T. Liu, "Sponsored search auctions: Recent advances and future directions," *ACM Trans. Intell. Syst. Technol.*, vol. 5, no. 4, pp. 60:1–60:34, 2014.

[350] S. Hofstatter, N. Craswell, B. Mitra, H. Zamani, and A. Hanbury, "Are we there yet? a decision framework for replacing term based retrieval with dense retrieval systems," *ArXiv*, vol. abs/2206.12993, 2022.

[351] A. Gionis, P. Indyk, R. Motwani *et al.*, "Similarity search in high dimensions via hashing," in *Vldb*, vol. 99, no. 6, 1999, pp. 518–529.