A Survey for Efficient Open Domain Question Answering

Qin Zhang[1], **Shangsi Chen**[1], **Dongkuan Xu**[2], **Qingqing Cao**[3],
Xiaojun Chen[1], **Trevor Cohn**[4], **Meng Fang**[5*]
[1]Shenzhen University; [2]North Carolina State University; [3]University of Washington;
[4]The University of Melbourne; [5]University of Liverpool
{qinzhang@, chenshangsi2021@email., xjchen@}szu.edu.cn; dxu27@ncsu.edu; qicao@cs.washington.edu;
tcohn@unimelb.edu.au; Meng.Fang@liverpool.ac.uk

Abstract

Open domain question answering (ODQA) is a longstanding task aimed at answering factual questions from a large knowledge corpus without any explicit evidence in natural language processing (NLP). Recent works have predominantly focused on improving the answering accuracy and have achieved promising progress. However, higher accuracy often requires more memory consumption and inference latency, which might not necessarily be efficient enough for direct deployment in the real world. Thus, a trade-off between accuracy, memory consumption and processing speed is pursued. In this paper, we will survey recent advancements in the efficiency of ODQA models and conclude core techniques for achieving efficiency. Additionally, we will provide a quantitative analysis of memory cost, query speed, accuracy, and overall performance comparison. Our goal is to keep scholars informed of the latest advancements and open challenges in ODQA efficiency research and contribute to the further development of ODQA efficiency.

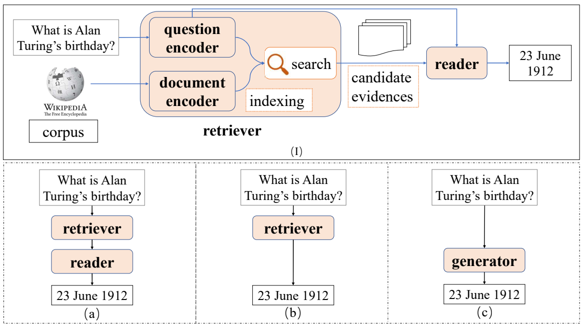


Figure 1: The general pipeline of ODQA models is shown in (I), along with three different ODQA frameworks: Retriever-Reader (a), Retriever-Only (b), and Generator-Only (c).

1 Introduction

Open domain question answering (ODQA) (Voorhees and Tice, 2000) is a longstanding task in natural language processing that can answer factoid questions, from a large knowledge corpus such as Wikipedia (Wikipedia, 2004) or BookCorpus (Zhu et al., 2015). Traditional QA models rely on explicit evidence texts to locate the answer, while ODQA models require the processing of large amounts of knowledge quickly to answer input questions. And compared to search engines, ODQA models aim to enhance user-friendliness by presenting the final answer to a question directly, rather than returning a list of relevant snippets or hyperlinks (Zhu et al., 2021).

Recently, ODQA systems have attracted considerable research attention and a classic framework of the ODQA system is implemented by encompassing an information retriever (IR) and a reader, i.e., *Retriever-Reader* (Chen et al., 2017). The task of IR is to retrieve evidence pieces from a large knowledge corpus. Popularly used IR can be TF-IDF (Chen et al., 2017), BM25 (Mao et al., 2021) and DPR (Karpukhin et al., 2020), etc. The target of the reader is understanding and reasoning the retrieved evidence to yield the answer. It is often achieved by transformer-based language models, such as BERT (Devlin et al., 2019), ALBERT (Lan et al., 2019) or generator T5 (Raffel et al., 2020), BART (Lewis et al., 2020a), GPT (Brown et al., 2020), etc. This two-module system enjoys a broad range of applications (Zhu et al., 2021).

However, most general-purpose ODQA models are computationally intensive, slow to infer, and expensive to train. One of the reasons for this is the huge index/document size (see Table 1 in Appendix B). For example, Karpukhin et al. (2020) processed an English Wikipedia corpus including 26 million articles and built a dense index with a size of 65GB. Besides, the majority of general-purpose ODQA models are developed with large pre-trained language models, which often contain millions of parameters. For instance, the state-of-the-art ODQA models on the Natural Question

*Corresponding author

dataset, R2-D2 (Fajcik et al., 2021) and UnitedQA (Cheng et al., 2021) have 1.29 billion and 2.09 billion model parameters, respectively. Storing the corpus index and pre-trained language models is memory-intensive (Xia et al., 2022) while evidence retrieving and reading are memory and time consuming. These make general-purpose ODQA models a big challenge for real-time use (Seo et al., 2019), such as on a mobile phone.

Towards this challenge, there are various trade-offs in building ODQA models that meet real-world application needs, such as the trade-offs among accuracy, memory consumption, inference speed, and so on (Izacard et al., 2020; Wu et al., 2020; Mao et al., 2021). NeurIPS 2020 organized an EfficientQA Competition (Min et al., 2021), aiming to build ODQA systems that can predict correct answers while also satisfying strict on-disk memory budgets. For this purpose, a line of work focused on building more efficient protocols. Besides Retriever-Reader, Retriever-Only (Lee et al., 2021b), Generator-Only (Roberts et al., 2020) are newly proposed protocols (see Fig. 1). Various efficiency techniques are also developed, such as index downsizing (Yamada et al., 2021; Lewis et al., 2022), fast searching (Lewis et al., 2021; Malkov and Yashunin, 2020), evidence retrieval or reading omitting (Roberts et al., 2020; Seonwoo et al., 2022; Lee et al., 2021b) and model size reducing (Yang and Seo, 2021; Singh et al., 2021) etc.

In this survey, we provide a comprehensive introduction to the broad range of methods that aim to improve efficiency with a focus on ODQA task [1]. In Section 2, we overview general-purpose ODQA models and discuss their strategies and limitations in terms of efficiency. In Section 3, we first walk through the key ODQA models which concentrate on efficiency, then conclude the core techniques used. Section 4 gives a quantitative analysis with an overall comparison of different frameworks and three specific aspects, i.e., memory cost, processing speed, and accuracy. Finally, in Section 5, we discuss the challenges reminded followed by the conclusion given in Section 6.

2 Overview of ODQA models

In this section, we summarize ODQA models into three typical frameworks (see in Fig. 1): Retriever-Reader, Retriever-Only, and Generator-Only. As described in Section 1, Retriever-Reader models include two modules: a retriever and a reader. For retrievers, traditional non-neural methods, such as TF-IDF (Chen et al., 2017) and BM25 (Mao et al., 2021), use sparse representations to measure term matching between questions and passages. However, these approaches can only capture lexical information, limiting capabilities in matching questions and passages (Qu et al., 2021). Differently, recent neural network-based dual-encoder retrievers (Karpukhin et al., 2020) encode questions and documents into a latent dense vector space where text semantics beyond terms can be adequately learned and measured. For readers, considering the way of obtaining answers, there exist two categories: extractive readers and generative readers. Extractive readers normally answer the question using a span from the context and the goal is to classify the start and end positions of the answer in the retrieved evidence (Karpukhin et al., 2020; Qu et al., 2021). And generative readers are not restricted to the input context and freely generate answers by autoregressively predicting tokens (Raffel et al., 2020; Izacard and Grave, 2021). Distinctively, Retriever-Only models only use one retriever to extract answers directly from a phrase or QA-pair knowledge base. And Generator-Only models directly generate answers with the question, not involving evidence retrieval and reading (Lee et al., 2021c; Lewis et al., 2021).

Retriever-Reader ODQA methods generally obtain good performance. However, due to dense encoding for corpus passages and longer evidence for answer reasoning, they normally suffer from a larger index size and a slower processing speed. In addition, the dual-encoder retrievers like DPR, encoding for questions and documents independently, ignored interaction between them and limited the retrieval performance (Khattab et al., 2021; Lu et al., 2022). In Retriever-Only ODQA models, the omission of the reading/generating step greatly improves the speed of answering questions. But there are a few limitations for Retriever-Only ODQA models: (1) lower performance on average compared to Retriever-Reader ODQA models since less information is considered during answer inference; (2) high storage requirement in terms of indexes for fine-grained retrieval units such as phrases or QA pairs. For Generator-Only ODQA models, skipping evidence retrieving and reading makes low memory costs and short processing time than two-stage systems. However, the performances of

[1] we compare with existing review works in Appendix A.

Generator-Only ODQA methods have much room for improvement. Additionally, real-world knowledge is updated routinely, and the huge training cost of the generative language models makes it laborious and impractical to keep them always up-to-date or retrain them frequently. Billions of parameters also make them storage-unfriendly and hard to apply on resource-constrained devices (Roberts et al., 2020). A diagram is provided with the typology of ODQA methods in Fig. 4 in the Appendix, while their main concerns are also indicated [2].

3 Efficient ODQA Models and Techniques

In this section, we first walk through the key ODQA models which concentrate on efficiency, and discuss their strengths and weaknesses as well as their unique characteristics in Section 3.1. Then we conclude the core techniques used in these models for improving the efficiency of ODQA, from data and model perspectives, respectively, in Section 3.2.

Before we start, we first take DPR on the Natural Questions (NQ) test dataset as an example to show the time each module needs during inference and their detailed memory costs in Fig. 2. We can see the total processing time DPR needs is 0.91 seconds (s)[3] where the inference speed is mainly affected by evidence searching (74.79%) and reading (23.95%). The total memory cost of DPR is 79.32GB which is huge. The index takes up 81.95% of the memory, the raw corpus takes 16.39% space, and the remaining 1.66% are for the models where the retriever model is around twice the size of the reader model.

Based on these observations, how to improve the efficiency of ODQA models focuses on the reduction of processing time and memory cost. To reduce processing time, we can accelerate evidence searching and reading. To reduce the memory cost, we can reduce the size of the index and model. Besides, some emerging directions are also proposed, such as jumping the retrieval part to generate answers using questions directly or retrieving answers directly to omit evidence reading. We introduce the details below.

3.1 Walk through Efficiency ODQA Models

In this subsection, we delve into the details of efficiency ODQA models. We categorize them into three classes regarding the different means of implementing efficiency, i.e., reducing processing time, reducing memory cost, and blazing new directions.

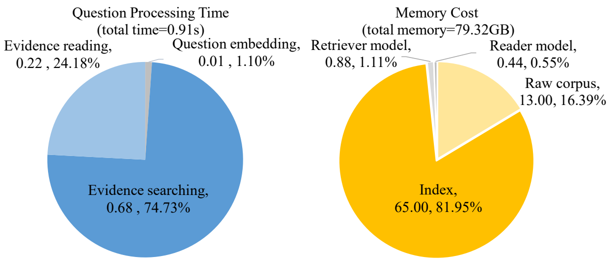


Figure 2: Query processing time and memory for DPR on NQ test set. We test them on an Nvidia GeForce Rtx 2080 Ti GPU and show the average results of 1000 examples.

3.1.1 Reducing Processing Time

When giving a question, the processing time for ODQA involves three stages: question embedding, evidence searching, and evidence reading. Whereas evidence searching and evidence reading occupy most of the processing time, researchers mainly focus on narrowing the time cost of the two stages.

By Accelerating Evidence Searching. Other than the traditional brute search method (Zhan et al., 2021), hierarchical navigable small world graphs (HNSW) (Malkov and Yashunin, 2020) and the approximate nearest neighbor (ANN) search (Johnson et al., 2021) techniques become increasingly popular, due to the characteristic of fast searching.

DPR (Yamada et al., 2021) and RePAQ (Lewis et al., 2021) adopt HNSW to achieve much faster search without a significant decline in retrieval accuracy. However, the negative effect HNSW brings is a larger index. For example, DPR with HNSW increases the index from 65GB to 151GB (Yamada et al., 2021). Besides, Locality Sensitive Hashing (LSH) (Neyshabur and Srebro, 2015) and Inverted File (IVF) (Sivic and Zisserman, 2003) are both efficient ANN methods to speedup search (Yamada et al., 2021; Lewis et al., 2022), but they often lead to a significant drop of retrieval accuracy (Yamada et al., 2021; Lewis et al., 2021, 2022). Concretely, LSH generates the same hashkey for similar embeddings through suitable hash functions, and then evidence retrieval is based on hashkeys (Wang et al., 2022). IVF constructs two-level indices using the K-means clustering method (Lewis et al., 2022). Different from LSH which can reduce the index size, IVF does not achieve this goal. Compared to LSH and IVF, Learned Index for large-scale DEnse passage Retrieval (LIDER) (Wang et al.,

[2] We collected available codes of the methods discussed in this paper and give their hyperlinks for convenience: https://github.com/hyintell/EfficientODQA.

[3] The passages in the corpus are embedded offline.

2022) makes a trade-off between search speed and retrieval accuracy through dynamically learning a corpus index when training. It achieves a faster search with a fewer drop in retrieval accuracy compared IVF, by predicting the location from the learned key-location distribution of the dataset. Specifically, LIDER builds two-level indices with a similar method IVF uses. LIDER further maps the documents in indices into hashkeys using the LSH method and sorts them based on the hashkeys. Meanwhile, the hashkeys are also used to train a multi-layer linear regression model for the location prediction of a hashkey in the sorted indexes. During inference, with a query embedded by DPR (Karpukhin et al., 2020), LIDER first calculates its hashkey, and finds its c nearest centroids. With these centroids, LIDER then searches the top-p nearest evidence in each subset in parallel. Finally, it merges all the retrieved evidence and selects the top-k ones as output. To conclude, LIDER is a powerful, efficient, and practical method for ODQA evidence searching.

By Accelerating Evidence Reading. Accelerating the evidence reading is another effective way to speed up the question processing of ODQA models. Actually, in the retrieved evidence, a high percentage of content is not pertinent to answers (Min et al., 2018). However, the reader module still allocated the same computational volume to these contents, which involves many unnecessary computations and prolongs the inference latency (Wu et al., 2020). Thus, the jumping reading strategy is proposed and studies have found it can bring certain inference speedup (Wu et al., 2020; Guan et al., 2022). Concretely, the jumping reading strategy dynamically identifies less relevant text blocks at each layer of computation by calculating an important score for each text block. Toward blocks with low scores, they will not be further involved in the subsequent processing.

Adaptive computation (AC) (Bengio et al., 2015; Graves, 2016) and Block-Skim (Guan et al., 2022) are efficient methods to ameliorate the reading efficiency following jumping reading strategy which manipulates the allocation of computation of the model input (Wu et al., 2020, 2021). SkyLineBuilder (Wu et al., 2020) applies AC to an extractive reader and dynamically decides which passage to allocate computation at each layer during reading. Further, Adaptive Passage Encoder (APE) (Wu et al., 2021) considers applying the AC strategy to Fusion-in-Decoder (FiD) system. In APE, the AC strategy is used to early stop the encoder of the generator to read the evidence that is less likely to include answers. Meanwhile, inspired by the idea of passage filtering before retrieval (Yang and Seo, 2021), Block-Skim (Guan et al., 2022) is proposed which skips question-irrelevant text blocks to optimize the reading speed. It first slices an input sequence into text blocks with a fixed length. A CNN module is utilized to compute the importance score for each block in each transformer layer, then the unimportant blocks are skipped. Block-Skim implements an average of 2.56 times speedup inference than BERT-based models with little loss of accuracy on multiple extractive QA datasets. This enlightens us that all BERT-based Retriever-Reader ODQA models can be optimized by Block-skim to speed up their inference.

3.1.2 Reducing Memory Cost

For ODQA models, there are three kinds of memory cost: index, model, and raw corpus. Normally, reducing the sizes of the index and model are two ways to break through and to achieve storage efficiency, while reducing raw corpus size results in certain knowledge source loss and a significant drop in performance (Yang and Seo, 2021).

By Reducing Index Size. The index of a corpus takes a major proportion of memory cost during running an ODQA system. The evidence-searching module, which is strongly related to the index size, is also the module that takes the most time during reference. Thus, downsizing the index is key to improving the efficiency of ODQA models. A line of research has tried to achieve this goal.

BPR (Yamada et al., 2021) and DrBoost (Lewis et al., 2022) are representative works in this direction. BPR reduces the index size by sacrificing data precision while DrBoost achieves this through compacting embedding dimension (Lewis et al., 2022). Specifically, BPR (Yamada et al., 2021) leverages a learning-to-hash technique (Cao et al., 2017; Wang et al., 2018) to hash continuous passage vectors into compact binary codes, which is different from DPR (Karpukhin et al., 2020) utilizing dense continuous embeddings of corpus passages. It optimizes the search efficiency of the retriever while maintaining accuracy through multi-target joint learning: evidence retrieval and reranking. During retrieval, top-c passages are retrieved with the Hamming distance of the binary codes. Then, the retrieved evidences are reranked with maximum inner product search (MIPS) (Shrivastava and Li, 2014; Guo et al., 2016) between the query dense vector and the passage binary codes. Finally, the top-k evidences are outputted, where k is much smaller than c. Differently, DrBoost (Lewis et al., 2022), a dense retrieval ensemble method inspired by boosting (Freund and Schapire, 1997), incrementally compacts the dimension of representations during training. Concretely, it builds sequentially multiple weak learners and integrates them into one stronger learner. Each weak learner consists of a BERT-based dual-encoder for encoding passages and questions by learning embeddings in low dimensions, normally 32-dim. The weak learners are trained iteratively using hard negative samples. The final embeddings for passages and questions are a linear combination of embeddings from all weak learners. Thus the dimension of the final embedding can be controlled by the iterative rounds during training, which makes the total embedding dimension flexible and the index size adjustable. One limitation of DrBoost is that it must keep multiple encoders simultaneously to compute the final representation of the question during inferring. To remedy this issue, DrBoost further distills all R question encoders (32 dim) into a single encoder (32*R dim). Therefore, the single encoder outputs the final question embedding directly, which achieves the goal of low resources.

By Reducing Model Size. Besides downsizing the index, compressing model is another way to cut the memory cost of ODQA systems. One way to accomplish this goal is building a comprehensive model to implement retrieval and reading simultaneously, instead of multiple models in traditional ODQA systems.

YONO (You Only Need One model) (Lee et al., 2021a) is a representative model in this way, which integrates retriever, reranker, and generator models into a T5-large based singular transformer pipeline. In this way, YONO achieves a less than 2GB model size which is as large as EMDR2 (Singh et al., 2021), and a higher QA performance. This makes YONO the best performance among models that are under the size of 2GB. Moreover, YONO can further manipulate its model size by adding or removing certain layers flexibly. To be specific, YONO first discards 18 decoder layers of the T5-large model and splits the rest model into four parts. The first 12 layers are for evidence retrieval; the middle 4 layers are for evidence reranking; the following 8 layers are for impressive encoding and the last 6 layers are for decoding. The hidden representations are progressively improved along the pipeline. A fully end-to-end training over all stages is performed to make full use of the capability of all modules. However, YONO still needs to do evidence indexing and searching, which is time-consuming. Thus, how to improve the processing speed of YONO is still a problem that needs to be solved urgently.

3.1.3 One-stage Frameworks

Besides the methods which accelerate evidence searching and reading and the methods that reduce the size of the index and model, some one-stage frameworks are proposed as well, such as generating the answer using the input question directly or retrieving answers directly from a finer-grained knowledge base (ie., phrases or question-answer pairs).

Directly Generate Answers. Some researchers blazed a brand new path that omits the whole evidence retrieval process, including corpus indexing and evidence searching, by leveraging generative language models (such as T5, BART, GPT) to tackle ODQA tasks (Roberts et al., 2020; Brown et al., 2020; Lewis et al., 2020a). Generative models have learned and stored the knowledge of a large-size corpus. Given a question, they can generate the answers directly. Without the evidence retrieval process, they save much processing time during ODQA, making them inference efficient. The main advantage of Generator-Only methods is that they can answer open-domain questions without any access to external knowledge (Roberts et al., 2020). And they output the literal text of the answer in a more free-form fashion. However, generally, there is a significant gap in QA performance between generative models and Retriever-Reader ODQA models, as well as the adequacy of explanation. Thus, single generator-based ODQA models are further combined with existing evidence retriever models (Lewis et al., 2020b; Izacard and Grave, 2021; Singh et al., 2021) to obtain better QA performance.

Directly Retrieve Answers. As discussed in the first few paragraphs of Section 3, evidence reading takes non-negligible processing time. An innovative idea to improve the efficiency of ODQA is to omit evidence reading. Without evidence reading, the document corpus is first preprocessed into a knowledge base offline. When encountering a new sample, the model searches the final answer from the knowledge base for the question directly (Seo et al., 2019; Lee et al., 2021b; Lewis et al., 2021).

RePAQ (Lewis et al., 2021) is representative of this framework. It first converts a large corpus to a knowledge base of question-answer (QA) pairs using a question generation model, then uses a lightweight QA-pair retriever to answer the questions. When inferring, it first calculates the similarity between the input question and each one in the knowledge base using the maximum inner product search (MIPS) technique (Shrivastava and Li, 2014; Guo et al., 2016), to retrieve the most similar QA pairs. The answer to the most similar question is returned as the output answer directly. However, the 220GB index for the 65 million QA pairs becomes a major drawback for RePAQ. Similarly, phrase-based ODQA models, such as DenSPI (Seo et al., 2019) and DensePhrases (Lee et al., 2021b), split the corpus documents into fine-grained phrases. They build an index for these phrases which can be retrieved directly as the predicted answers. Similar to RePAQ, omitting evidence reading makes phrase-based ODQA models faster than Retriever-Reader ODQA models when processing questions, as analyzed in Section 4.3.

3.2 Core Techniques

This section concludes the core techniques commonly used in existing ODQA models with respect to improving efficiency. It can be briefly divided into two categories: data-based and model-based techniques. Data-based techniques mainly focus on the reduction of the index, which can be downsized from different hierarchies such as the number of corpus passages, feature dimension, and storage unit per dimension. Model-based techniques try to reduce the model size while avoiding a significant drop in performance. Model pruning and knowledge distillation are commonly used techniques.

3.2.1 Data-based techniques

Passage Filtering. Among the huge corpus ODQA models rely on, there are massive passages that contain little useful information and are unlikely to be evidence for answers. Thus, filtering unrelated passages is a way to reduce the memory cost of corpus without a large negative impact. For example, some researchers have designed a linear classifier to discriminate and discard unnecessary passages before evidence retrieval (Izacard et al., 2020; Yang and Seo, 2021).

Dimension Reduction. Another way to reduce the memory cost is to reduce the dimension for dense passage representations. To achieve this goal, Izacard et al. (2020) learns an additional feed-forward layer to project the high-dimensional embeddings to lower ones. Principle component analysis (PCA) is another efficient technique that is commonly used to reduce the dimension of passage representations without a loss of important information (Ma et al., 2021; Zouhar et al., 2022). In work Ma et al. (2021), PCA is used to build a projection matrix to project the raw data onto the principal components using an orthonormal basis.

Product Quantization. Product quantization (PQ) (Jégou et al., 2011) further reduces the index size by reducing the storage cost of each dimension of the embeddings. It divides a d-dimensional vector into n sub-vectors with d/n dimension and quantifies these sub-vectors independently using k-means (Izacard et al., 2020; Ma et al., 2021; Yang and Seo, 2021). However, PQ also results in a significant drop in accuracy while it reduces the index size.

The three techniques introduced above are adopted jointly in Fusion-in-Decoder with Knowledge Distillation (FiD-KD) (Izacard et al., 2020) to reduce the memory cost of one ODQA system. It obtains competitive performance compared to the original system while compressing memory from more than 70GB to less than 6GB.

3.2.2 Model-based techniques

Model Pruning. Most recent works on open domain question answering (Chen et al., 2017; Guu et al., 2020) prefer to adopt large pre-trained language models (Devlin et al., 2019; Raffel et al., 2020) as passage retriever, reader or generator due to their powerful deep semantic understanding capability. These large models have millions or even billions of parameters, requiring large storage, long training time, and leading to slow inference. To this point, some researchers have turned to adopt more lightweight language models (Yang and Seo, 2021). For example, a smaller pre-trained language model, MobileBERT (Sun et al., 2020), has been used to reduce the size of an ODQA system to 972MB (Yang and Seo, 2021). Parameter sharing is another way to constrain the model size. Skylinebuilder (Wu et al., 2020) and RePAQ downsize their model by using the parameter sharing encoders, i.e., ALBERT (Lan et al., 2019). More lightweight pre-trained language models have been proposed and verified in other natural language tasks, such as machine reading comprehension (Fan et al., 2019; Sajjad et al., 2020; Lagunas et al., 2021; Xia et al., 2022). They obtain smaller model sizes and achieve high accuracy for downstream tasks, including ODQA tasks.

Knowledge Distillation. Compared to structure pruning, knowledge distillation pays more attention to effectively improving question processing speed. Knowledge distillation, which transfers knowledge from a large model into a small one, has been widely used in several NLP tasks, including ODQA and MRC tasks (Sanh et al., 2019; Sun et al., 2020; Izacard and Grave, 2020; Lewis et al., 2022; Yang and Seo, 2021). For example, Minimal R&R system (Yang and Seo, 2021) and DrBoost (Lewis et al., 2022) both integrate multiple modules into a single one via knowledge distillation.

4 Quantitative Analysis

This section gives a quantitative analysis of the aforementioned ODQA models. We first give an overall comparison of different frameworks and further discuss the methods quantitatively from three specific aspects: memory cost, processing speed, and accuracy[4]. At the end of the analysis, the following subsection summarizes and concludes what has been analyzed and discussed.

4.1 Overall Comparison

In Table 1 in Appendix B, we demonstrate a comprehensive comparison of efficiency-related ODQA models from three aspects: memory cost, processing speed, and answering quality. Specifically, total memory storage, detailed model size, and index size are listed to show details of memory cost. The number of questions that can be answered per second (Q/s) demonstrates the processing speed. Exact match (EM) scores on Natural Questions (Kwiatkowski et al., 2019) and TriviaQA (Joshi et al., 2017) datasets indicate answering quality.

Concerning comparison between different frameworks, we can see that two-stage methods (Retriever-Reader) generally obtain better ODQA **performances** than one-stage methods (i.e., Retriever-Only and Generator-Only). The best end-to-end EM performance on NQ (55.9%) and TriviaQA (74.8%) datasets are obtained by R2-D2+reranker and GAR_extractive respectively. They are both under the Retriever-Reader framework. The second-best ODQA performances on NQ (54.7%) and TriviaQA (72.1%) are obtained by UnitedQA and Fid-large+KD_DPR methods, which are also under the two-stage frameworks.

In terms of total **memory cost**, i.e., the sum of model size and the index size, Generator-Only systems keep generally low memory overhead. Except GPT-3, the rest of the Generator-Only systems take less than 50GB of memory, and five methods out of the eight are less than 5GB. On the contrary, most Retriever-Only ODQA models require huge memory, normally greater than 200GB. The method DenSPI needs a 2002.69GB memory cost, which is enormous. Retriever-Reader ODQA models have a wide range in terms of memory cost, from 0.31GB to 363.26GB. Overall speaking, Minimal R&R achieves the smallest memory overhead (0.31GB) while DenSPI keeps the largest one (2002.69GB).

In terms of **processing speed**, which determines how fast one ODQA system can answer a given question, one-stage methods generally achieve higher processing speed than two-stage methods, especially Retriever-Only systems. Among the eight Retriever-Only methods, five of them can process more than 20 questions per second (Q/s) and RePAQ_XL and RePQA_base can answer 800 and 1400 questions per second respectively, which is impressive. For the methods with slow processing speed, Fig-large and RAG-seq from the Retriever-Reader framework are the two slowest systems, which process less than 1 question per second.

To conclude, Fig. 3 gives a visual presentation for **comprehensive comparison** of efficiency-related ODQA models according to different frameworks. By using the NQ evaluation dataset as an example, it illustrates the detailed model size, index size, EM scores, and processing speed respectively. From Fig. 3, we can see each framework has its strengths and weaknesses. Retriever-Only systems achieve significantly high processing speeds but cost enormous memory storage. Generator-Only systems require the least memory storage. However, the main concern of them is the answering quality while the majority of these systems' EM scores are less than 30% on NQ datasets. Two-stage Retriever-Reader systems relatively behave balanced. They achieve high EM scores and obtain moderate memory cost and processing speed.

[4] We summarize the corpus and the related partition and segmentation strategies, as well as metrics introduction in Appendix B

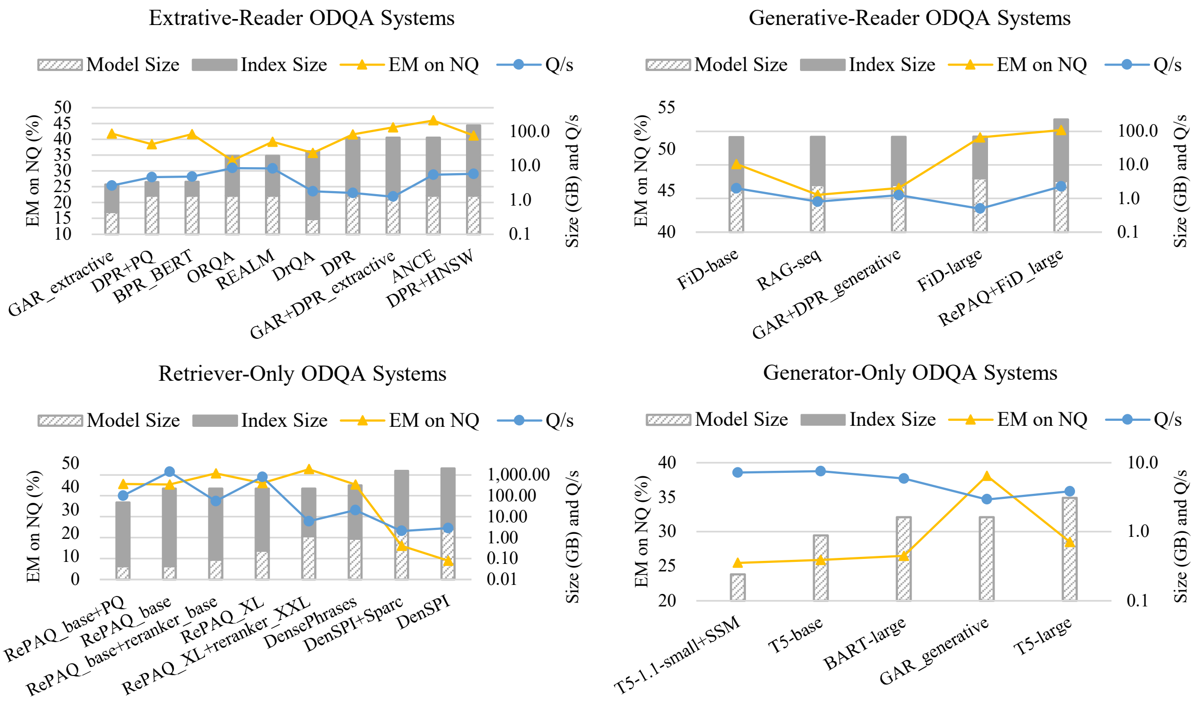


Figure 3: Comprehensive comparison of ODQA models in terms of memory cost, processing speed, and EM accuracy on NQ evaluation dataset. The extractive-reader and generative-reader ODQA systems both belong to Retriever-Reader ODQA systems.

4.2 Details in Memory Cost

The total memory cost depends on the model size and the index size.

Index Size. For the index size, the two kinds of one-stage frameworks are two extremes. Generator-Only methods do not require creating an index file while Retriever-Only methods generally need a huge storage space for index. Most two-stage methods have a moderate index of 65GB or less.

For Retriever-Reader ODQA systems, the 65GB index set of dense passage embedding, developed by DPR (Karpukhin et al., 2020), is the most commonly adopted index set. It is adopted by 17 methods as we listed in Table 1 in Appendix B. Based on this index set, DrQA and GAR_extractive represent passages into sparse vectors, obtained a much smaller index size (26GB) (Chen et al., 2017; Mao et al., 2021). Through the product quantization (PQ) technique, DPR+PQ compresses the index size from 65GB to 2GB and the index size of RePAQ is from 220GB to 48GB. On the other side, BPR (Yamada et al., 2021) creates a small index of less than 2.1GB. It also improves the answering performance from 41.6% to 49% on the NQ dataset by replacing the BERT-based reader with the ELECTRA-large reader. Meanwhile, Minimal R&R (Yang and Seo, 2021) establishes the smallest index of less than 0.15GB with a price of a significant drop in ODQA performance has been paid.

For Retriever-Only ODQA systems, DenSPI+Sparc (Lee et al., 2020) and DensePhrase (Lee et al., 2021b) smallen the phrase index by pointer sharing, phrase filtering, and PQ. However, the phrase index is still larger than 1000GB. DensePhrases further cuts down the index to 320GB by omitting sparse representations and using SpanBERT-based encoder while a relatively high performance remained. SpanBERT-based represents phrases into a lower-dimension space (Joshi et al., 2020) than that in DenSPI+Sparc. And DrBoost (Lewis et al., 2022) builds an index under 1GB where a passage is represented with a 190-dim vector through the boosting and PQ techniques.

Model Size The model size involves all modules present in one ODQA system, including the retriever and the reader. It has a great range, from 0.04GB to 700GB. Among all mentioned ODQA models, a quarter of ones have model sizes less than 1GB; the model sizes of 40% systems are between 1∼2GB and 12.5% ones have sizes between 2∼3GB; 7.5% systems have model sizes between 3∼4GB; the remaining 15% models weigh larger than 4GB. Specifically, GPT-3 (Brown et al., 2020) has an extremely huge model size of 700GB. Besides it, another three systems are obtaining relatively large models: T5-1.1-XL_SSM (45.27GB) (Roberts et al., 2020), UnitedQA (8.36GB) (Cheng et al., 2021) and R2-D2+reranker (5.16GB) (Fajcik et al., 2021), while the system with the smallest model (0.04GB) is achieved by RePAQ-base (Lewis et al., 2021). Specifically, GPT-3 keeps the largest model (700GB) and achieves relatively high performance, i.e., 71.2% EM on TriviaQA (top 1) and 29.9% EM on NQ dataset (top 3), compared to other models with the same Generator-Only framework. Minimal R&R (Yang and Seo, 2021) cuts down the total model size to 0.17GB. DrQA (Chen et al., 2017) has a small total model size 0.27GB in that its retriever is non-parameter BM25 and the reader relies on LSTM with fewer parameters. GAR_extractive (Mao et al., 2021) maintains a small total model size and achieves the best performance on TriviaQA (74.8%) and similar performance with DPR on NQ (41.8%). RePAQ (Lewis et al., 2021) achieves the smallest model of 0.04GB. It also gains competitive performance compared to DPR.

Most ODQA models are implemented with PLMs that are less than 2GB. A few ODQA models keep the total model size more than 3GB to achieve higher performance, like FiD-large+KD_DPR (Izacard and Grave, 2020), RePAQ+FiD_large (Lewis et al., 2021), UnitedQA (Cheng et al., 2021) and R2-D2_reranker (Fajcik et al., 2021). As they employ either larger or more pre-trained language models to focus on improving performance.

4.3 Details on Latency

In terms of latency, i.e., processing speed, most ODQA models answer less than 10 questions per second. Retriever-Only ODQA models bring faster processing speed than the other three frameworks. Compared to phrase-base systems, the QA-pair-based system RePAQ (Lewis et al., 2021) and its variants win the fastest inference speed among the listed ODQA models, up to 1400 Q/s. Generator-Only ODQA models also achieve higher Q/s than Retriever-Reader ODQA models, as they do not need retrieving evidence from a larger corpus which is time-consuming.

5 Discussion

In this section, we summarize and illustrate the insights and future directions from the following aspects. We first summarize the key points to improve the effectiveness of ODQA systems, from the two aspects of index and model respectively. In terms of index size, it is worth exploring deeper on generative models and the techniques of compacting embedding. In terms of model size, knowledge distillation is a promising direction to reduce model size while another direction is the application of lightweight models. In addition, one-stage ODQA models are also worthy of research.

Additionally, we provide some advice on model recommendations under different requirements. For example, if we pursue real-time feedback, Retriever-Only systems should be good choices; if we are limited by computing resources, Generator-Only systems are suitable candidates; and if we need to trade off performance, memory cost and processing time, Retriever-Reader systems are relatively more appropriate. In general, for researchers who are interested in improving the state-of-the-art efficiency methods on ODQA tasks, this survey can serve as an entry point to find opportunities for new research directions.

However, some salient challenges need to be addressed in the way of ODQA efficiency research. One of the worrisome things is that most ODQA approaches are computation-heavy and energy-expensive. How can the ODQA system be deployed in low-power devices with limited computing resources and mobile devices is still very challenging. Another thing is that it seems to be inadequate to evaluate the efficiency of ODQA models only on accuracy, memory, and processing time, due to many other factors that should be considered and traded off. For example, it is also important to establish what resource, e.g., money, time, and data for model training, power consumption, carbon emissions, etc.

6 Conclusion

In this survey, we retrospected the typical literature according to three different frameworks of open domain question answering (ODQA) systems. Further, we provided a broad overview of existing methods to increase efficiency for ODQA models and discussed their limitations. In addition, we performed a quantitative analysis in terms of efficiency and offered certain suggestions about method selections of open domain question answering. Finally, we discussed possible open challenges and potential future directions of efficient ODQA models.

7 Limitations

It seems to be difficult to evaluate the efficiency of ODQA models fairly and impartially due to multiple factors that should be considered and need to be traded off. On the one hand, it is not enough to only use accuracy, memory, and processing time to evaluate effectiveness. It is also important to establish what resource, e.g., money, power consumption, carbon emissions, etc., one attempt to constrain (Treviso et al., 2022). On the other hand, how to deploy models and what tools model implementation relies on contributes to inequity growth (Blodgett et al., 2020). It is extremely challenging to unify the deployment of all models and the tools they rely on and to achieve a truly fair and unbiased effectiveness comparison.

8 Ethics Statement

Our work focuses on summarizing and discussing the accuracy, inference speed, and memory cost of open domain question answering systems. We believe that our work is helpful for researchers who are interested in improving the state-of-the-art efficiency methods on ODQA tasks. We do not anticipate any ethical concerns arising from the research presented in this paper.

Acknowledgments

This research was supported by National Natural Science Foundation of China (62206179, 92270122), Guangdong Provincial Natural Science Foundation (2022A1515010129, 2023A1515012584), University stability support program of Shenzhen (20220811121315001), Shenzhen Research Foundation for Basic Research, China (JCYJ20210324093000002).

References

Emmanuel Bengio, Pierre-Luc Bacon, Joelle Pineau, and Doina Precup. 2015. Conditional computation in neural networks for faster models. CoRR, abs/1511.06297.

Su Lin Blodgett, Solon Barocas, Hal Daumé III, and Hanna Wallach. 2020. Language (technology) is power: A critical survey of "bias" in NLP. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 5454–5476, Online. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In Advances in Neural Information Processing Systems, volume 33, pages 1877–1901. Curran Associates, Inc.

Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S. Yu. 2017. Hashnet: Deep learning to hash by continuation. In Proceedings of the IEEE International Conference on Computer Vision (ICCV).

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Hao Cheng, Yelong Shen, Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2021. UnitedQA: A hybrid approach 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), pages 3080–3090, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Romina Etezadi and Mehrnoush Shamsfard. 2022. The state of the art in open domain complex question answering: a survey. Applied Intelligence, pages 1–21.

Martin Fajcik, Martin Docekal, Karel Ondrej, and Pavel Smrz. 2021. R2-D2: A modular baseline for open-domain question answering. In Findings of the Association for Computational Linguistics: EMNLP 2021, pages 854–870, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Angela Fan, Edouard Grave, and Armand Joulin. 2019. Reducing transformer depth on demand with structured dropout. CoRR, abs/1909.11556.

Yoav Freund and Robert E Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences, 55(1):119–139.

Alex Graves. 2016. Adaptive computation time for recurrent neural networks. ArXiv, abs/1603.08983.

Yue Guan, Zhengyi Li, Zhouhan Lin, Yuhao Zhu, Jingwen Leng, and Minyi Guo. 2022. Block-skim: Efficient question answering for transformer. Proceedings of the AAAI Conference on Artificial Intelligence, 36(10):10710–10719.

Jiafeng Guo, Yinqiong Cai, Yixing Fan, Fei Sun, Ruqing Zhang, and Xueqi Cheng. 2022. Semantic models for the first-stage retrieval: A comprehensive review. ACM Trans. Inf. Syst., 40(4).

Ruiqi Guo, Sanjiv Kumar, Krzysztof Choromanski, and David Simcha. 2016. Quantization based fast inner product search. In Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, volume 51 of Proceedings of Machine Learning Research, pages 482–490, Cadiz, Spain. PMLR.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In International Conference on Machine Learning, pages 3929–3938. PMLR.

Zhen Huang, Shiyi Xu, Minghao Hu, Xinyi Wang, Jinyan Qiu, Yongquan Fu, Yuncai Zhao, Yuxing Peng, and Changjian Wang. 2020. Recent trends in deep learning based open-domain textual question answering systems. IEEE Access, 8:94341–94356.

Gautier Izacard and Edouard Grave. 2020. Distilling knowledge from reader to retriever for question answering. CoRR, abs/2012.04584.

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

Gautier Izacard, Fabio Petroni, Lucas Hosseini, Nicola De Cao, Sebastian Riedel, and Edouard Grave. 2020. A memory efficient baseline for open domain question answering. CoRR, abs/2012.15156.

Hervé Jégou, Matthijs Douze, and Cordelia Schmid. 2011. Product quantization for nearest neighbor search. IEEE Transactions on Pattern Analysis and Machine Intelligence, 33:117–128.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-scale similarity search with gpus. IEEE Transactions on Big Data, 7(3):535–547.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving Pre-training by Representing and Predicting Spans. Transactions of the Association for Computational Linguistics, 8:64–77.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In Annual Meeting of the Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 6769–6781, Online. Association for Computational Linguistics.

Omar Khattab, Christopher Potts, and Matei Zaharia. 2021. Relevance-guided Supervision for OpenQA with ColBERT. Transactions of the Association for Computational Linguistics, 9:929–944.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. Transactions of the Association for Computational Linguistics, 7:453–466.

François Lagunas, Ella Charlaix, Victor Sanh, and Alexander Rush. 2021. Block pruning for faster transformers. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 10619–10629, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. CoRR, abs/1909.11942.

Haejun Lee, Akhil Kedia, Jongwon Lee, Ashwin Paranjape, Christopher D. Manning, and Kyoung-Gu Woo. 2021a. You only need one model for open-domain question answering. CoRR, abs/2112.07381.

Jinhyuk Lee, Minjoon Seo, Hannaneh Hajishirzi, and Jaewoo Kang. 2020. Contextualized sparse representations for real-time open-domain question answering. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 912–919, Online. Association for Computational Linguistics.

Jinhyuk Lee, Mujeen Sung, Jaewoo Kang, and Danqi Chen. 2021b. 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), pages 6634–6647, Online. Association for Computational Linguistics.

Jinhyuk Lee, Alexander Wettig, and Danqi Chen. 2021c. Phrase retrieval learns passage retrieval, too. In

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 3661–3672, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

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

Patrick Lewis, Barlas Oguz, Wenhan Xiong, Fabio Petroni, Scott Yih, and Sebastian Riedel. 2022. Boosted dense retriever. In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 3102–3117, Seattle, United States. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020b. Retrieval-augmented generation for knowledge-intensive nlp tasks. In Advances in Neural Information Processing Systems, volume 33, pages 9459–9474. Curran Associates, Inc.

Patrick Lewis, Yuxiang Wu, Linqing Liu, Pasquale Minervini, Heinrich Küttler, Aleksandra Piktus, Pontus Stenetorp, and Sebastian Riedel. 2021. PAQ: 65 Million Probably-Asked Questions and What You Can Do With Them. Transactions of the Association for Computational Linguistics, 9:1098–1115.

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

Xueguang Ma, Minghan Li, Kai Sun, Ji Xin, and Jimmy Lin. 2021. Simple and effective unsupervised redundancy elimination to compress dense vectors for passage retrieval. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 2854–2859, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. 2021. Generation-augmented retrieval 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), pages 4089–4100, Online. Association for Computational Linguistics.

Sewon Min, Jordan Boyd-Graber, Chris Alberti, Danqi Chen, Eunsol Choi, Michael Collins, Kelvin Guu, Hannaneh Hajishirzi, Kenton Lee, Jennimaria Palomaki, Colin Raffel, Adam Roberts, Tom Kwiatkowski, Patrick Lewis, Yuxiang Wu, Heinrich Küttler, Linqing Liu, Pasquale Minervini, Pontus Stenetorp, Sebastian Riedel, Sohee Yang, Minjoon Seo, Gautier Izacard, Fabio Petroni, Lucas Hosseini, Nicola De Cao, Edouard Grave, Ikuya Yamada, Sonse Shimaoka, Masatoshi Suzuki, Shumpei Miyawaki, Shun Sato, Ryo Takahashi, Jun Suzuki, Martin Fajcik, Martin Docekal, Karel Ondrej, Pavel Smrz, Hao Cheng, Yelong Shen, Xiaodong Liu, Pengcheng He, Weizhu Chen, Jianfeng Gao, Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Wen-tau Yih. 2021. Neurips 2020 efficientqa competition: Systems, analyses and lessons learned. In Proceedings of the NeurIPS 2020 Competition and Demonstration Track, volume 133 of Proceedings of Machine Learning Research, pages 86–111. PMLR.

Sewon Min, Victor Zhong, Richard Socher, and Caiming Xiong. 2018. Efficient and robust question answering from minimal context over documents. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1725–1735, Melbourne, Australia. Association for Computational Linguistics.

Behnam Neyshabur and Nathan Srebro. 2015. On symmetric and asymmetric lshs for inner product search. In Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37, ICML'15, page 1926–1934. JMLR.org.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. RocketQA: An optimized training approach to dense passage retrieval for open-domain question answering. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 5835–5847, Online. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res., 21(140):1–67.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 5418–5426, Online. Association for Computational Linguistics.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. 2020. Poor man's bert: Smaller and faster transformer models. ArXiv, abs/2004.03844.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. CoRR, abs/1910.01108.

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

Yeon Seonwoo, Juhee Son, Jiho Jin, Sang-Woo Lee, Ji-Hoon Kim, Jung-Woo Ha, and Alice Oh. 2022. Two-step question retrieval for open-domain QA. In Findings of the Association for Computational Linguistics: ACL 2022, pages 1487–1492, Dublin, Ireland. Association for Computational Linguistics.

Xiaoyu Shen, Svitlana Vakulenko, Marco Del Tredici, Gianni Barlacchi, Bill Byrne, and A. Gispert. 2022. Low-resource dense retrieval for open-domain question answering: A comprehensive survey. ArXiv, abs/2208.03197.

Anshumali Shrivastava and Ping Li. 2014. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In Advances in Neural Information Processing Systems, volume 27. Curran Associates, Inc.

Devendra Singh, Siva Reddy, Will Hamilton, Chris Dyer, and Dani Yogatama. 2021. End-to-end training of multi-document reader and retriever for open-domain question answering. In Advances in Neural Information Processing Systems, volume 34, pages 25968–25981. Curran Associates, Inc.

Sivic and Zisserman. 2003. Video google: a text retrieval approach to object matching in videos. In Proceedings Ninth IEEE International Conference on Computer Vision, pages 1470–1477 vol.2.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. MobileBERT: a compact task-agnostic BERT for resource-limited devices. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 2158–2170, Online. Association for Computational Linguistics.

Marcos Vinícius Treviso, Tianchu Ji, Ji-Ung Lee, Betty van Aken, Qingqing Cao, Manuel R. Ciosici, Michael Hassid, Kenneth Heafield, Sara Hooker, Pedro Henrique Martins, André F. T. Martins, Peter Milder, Colin Raffel, Edwin Simpson, Noam Slonim, Niranjan Balasubramanian, Leon Derczynski, and Roy Schwartz. 2022. Efficient methods for natural language processing: A survey. ArXiv, abs/2209.00099.

Ellen M. Voorhees and Dawn M. Tice. 2000. The TREC-8 question answering track. In Proceedings of the Second International Conference on Language Resources and Evaluation (LREC'00), Athens, Greece. European Language Resources Association (ELRA).

Jingdong Wang, Ting Zhang, jingkuan song, Nicu Sebe, and Heng Tao Shen. 2018. A survey on learning to hash. IEEE Transactions on Pattern Analysis and Machine Intelligence, 40(4):769–790.

Yifan Wang, Haodi Ma, and Daisy Zhe Wang. 2022. Lider: An efficient high-dimensional learned index for large-scale dense passage retrieval. ArXiv, abs/2205.00970.

Wikipedia. 2004. Wikipedia. PediaPress.

Yuxiang Wu, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2021. Training adaptive computation for open-domain question answering with computational constraints. 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), pages 447–453, Online. Association for Computational Linguistics.

Yuxiang Wu, Sebastian Riedel, Pasquale Minervini, and Pontus Stenetorp. 2020. Don't read too much into it: Adaptive computation for open-domain question answering. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 3029–3039, Online. Association for Computational Linguistics.

Mengzhou Xia, Zexuan Zhong, and Danqi Chen. 2022. Structured pruning learns compact and accurate models. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1513–1528, Dublin, Ireland. Association for Computational Linguistics.

Ikuya Yamada, Akari Asai, and Hannaneh Hajishirzi. 2021. Efficient passage retrieval with hashing for open-domain question answering. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pages 979–986, Online. Association for Computational Linguistics.

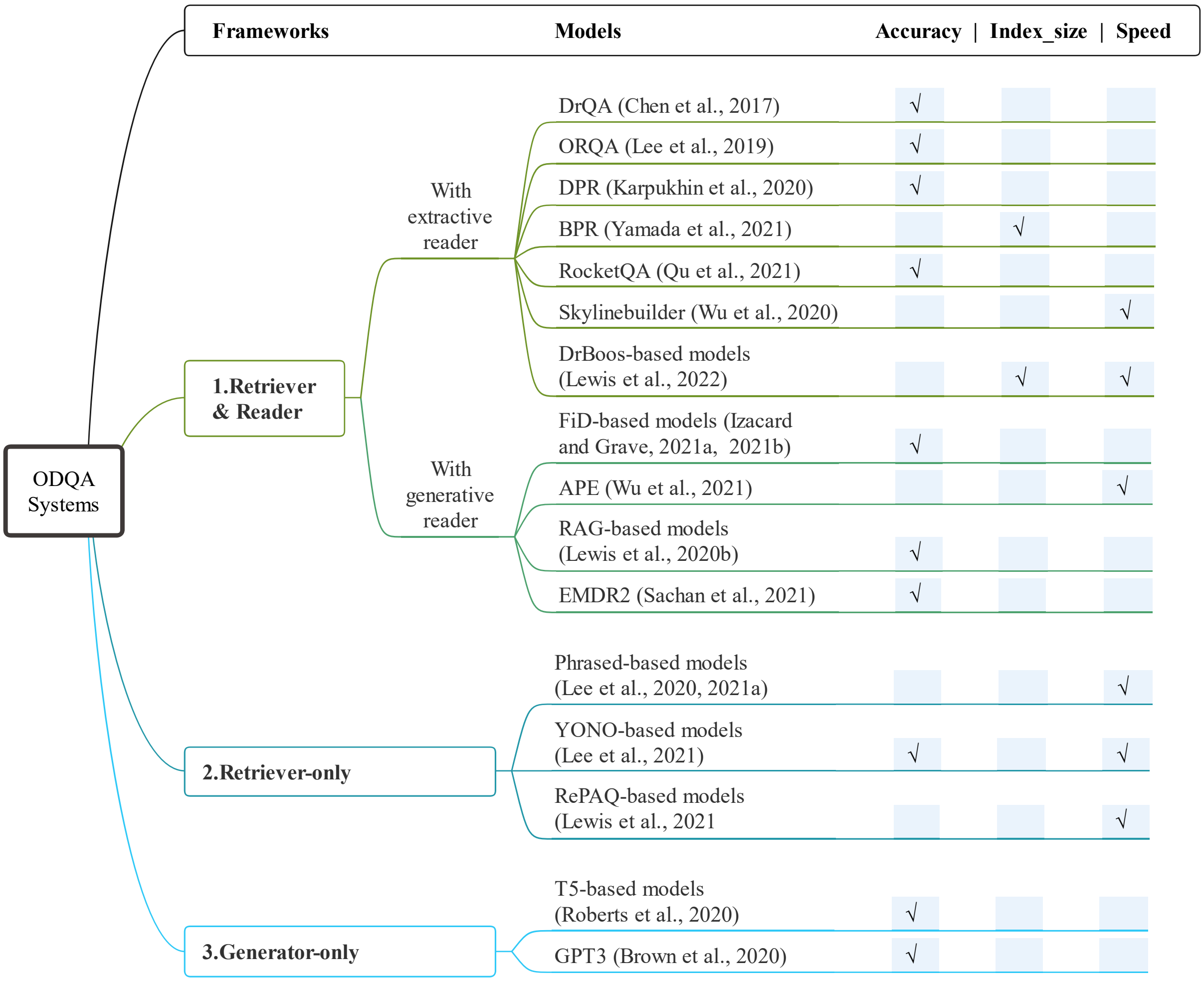
Sohee Yang and Minjoon Seo. 2021. Designing a minimal retrieve-and-read system for open-domain question answering. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 5856–5865, Online. Association for Computational Linguistics.

Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Jointly optimizing query encoder and product quantization to improve retrieval performance. In Proceedings of the 30th ACM International Conference on Information &amp; Knowledge Management, CIKM '21, page 2487–2496, New York, NY, USA. Association for Computing Machinery.

Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. 2021. Retrieving and reading: A comprehensive survey on open-domain question answering. CoRR, abs/2101.00774.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In 2015 IEEE International Conference on Computer Vision (ICCV), pages 19–27.

Vilém Zouhar, Marius Mosbach, Miaoran Zhang, and Dietrich Klakow. 2022. Knowledge base index compression via dimensionality and precision reduction. In Proceedings of the 1st Workshop on Semiparametric Methods in NLP: Decoupling Logic from Knowledge, pages 41–53, Dublin, Ireland and Online. Association for Computational Linguistics.



Frameworks		Models	Accuracy	Index_size	Speed
1.Retriever & Reader	With extractive reader	DrQA (Chen et al., 2017)	√		
		ORQA (Lee et al., 2019)	√		
		DPR (Karpukhin et al., 2020)	√		
		BPR (Yamada et al., 2021)		√	
		RocketQA (Qu et al., 2021)	√		
		Skylinebuilder (Wu et al., 2020)			√
		DrBoos-based models (Lewis et al., 2022)		√	√
	With generative reader	FiD-based models (Izacard and Grave, 2021a, 2021b)	√		
		APE (Wu et al., 2021)			√
		RAG-based models (Lewis et al., 2020b)	√		
		EMDR2 (Sachan et al., 2021)	√		
2.Retriever-only		Phrased-based models (Lee et al., 2020, 2021a)			√
		YONO-based models (Lee et al., 2021)	√		√
		RePAQ-based models (Lewis et al., 2021			√
3.Generator-only		T5-based models (Roberts et al., 2020)	√		
		GPT3 (Brown et al., 2020)	√		

Figure 4: Typology of ODQA systems and their main concerns in terms of accuracy, memory size, and processing speed. The symbol ✓in this figure demonstrates whether the corresponding models consider the concerns in terms of accuracy, index size, and speed respectively, or not.

A Connection to Existing Related Surveys

ODQA has been discussed and summarized with a broad overview of techniques for NLP in several survey papers. However, they more focus on deep neural models for improving ODQA performance. Specifically, the survey given by Huang et al. (2020) introduces deep learning-based ODQA models proposed in the early years, which are mainly based on LSTM or CNN. Modern Transformer-based ODQA models are not included. Work given by Zhu et al. (2021) provides a comprehensive literature review of ODQA models, with particular attention to techniques incorporating neural machine reading comprehension models. Guo et al. (2022) focuses on the semantic models of the first-stage retrieval models. Shen et al. (2022) pays more attention to how to train the dense retrievers effectively with fewer annotation training data. Treviso et al. (2022) retrospects the efficient methods in natural language processing (NLP). It mainly involves the upstream generic pre-trained language models and training methods. Etezadi and Shamsfard (2022) mainly concentrates on the comparison of ODQA methods for complex question answering. As far as we know, there is no survey summarizing ODQA methods from the efficiency perspective so far, which inspires us to overview the efficient ODQA models in this paper.

B Corpus and Metrics Normally Used

Corpus. The most commonly used corpus for open domain question answering systems is the 2018-12-20 dump of Wikipedia corpus, which contains 21 million 100-word-long passages after removing semi-structured data (tables, information boxes, lists, and the disambiguation pages) (Karpukhin et al., 2020). Most ODQA models, such as RocketQA (Qu et al., 2021), FiD (Izacard and Grave, 2021), and R2-D2 (Fajcik et al., 2021), directly

Table 1: Comprehensive analysis of memory cost (model size, index size, and the total size), processing speed, and EM accuracy on NQ. Results marked with symbol $*$ were obtained on an Nvidia GeForce Rtx 2080 Ti GPU over 100 examples.

Frameworks		Systems	Grouped by Memory Cost(GB)	Memory Cost(GB)			Processing Speed	EM score (%)	
				Models	Index	Total	Q/s	NQ	TriviaQA
Retriever-Reader	Extractive-Reader	Minimal R&R	(0, 10]	0.17	**0.15**	**0.31**	-	32.60	48.75
		SkylineBuilder		0.07	2.40	2.47	-	34.20	-
		BM25+BERT-base		0.44	2.40	2.84	4.68*	-	-
		GAR_extractive		0.44	2.40	2.84	2.61*	41.80	**74.80**
		DrBoost+PQ(8-dim)		2.64	0.42	3.06	-	-	-
		DPR+PQ		1.32	2.00	3.32	4.67*	38.40	52.00
		BPR_BERT		1.32	2.10	3.42	4.81*	41.60	56.80
		DrBoost+PQ(4-dim)		2.64	0.84	3.48	-	-	-
		BPR_ELECTRA-large		2.22	2.10	4.32	-	49.00	65.60
		DrBoost	(10, 50]	2.64	13.00	15.64	-	-	-
		ORQA		1.32	18.00	19.32	8.60	33.30	45.00
		REALM		1.32	18.00	19.32	8.40	39.20	-
		DrQA		0.27	26.00	26.27	1.80	35.70	-
		ColBERT-QA-base	(50, 100]	0.88	65.00	65.88	-	42.30	64.60
		ANCE		1.32	65.00	66.32	5.51*	46.00	57.50
		DPR		1.32	65.00	66.32	1.60*	41.50	56.80
		GAR+DPR_extractive		1.32	65.00	66.32	1.25*	43.80	-
		RocketQA		1.50	65.00	66.50	-	42.80	-
		ColBERT-QA-large		1.76	65.00	66.76	-	47.80	70.10
		ERNIE-Search_base		1.76	65.00	66.76	-	-	-
		R2-D2_reranker		5.16	65.00	70.16	-	**55.90**	69.90
		UnitedQA		8.36	65.00	73.36	-	54.70	70.50
		DPR+HNSW	(100, 500])	1.32	151.00	152.32	5.82*	41.20	56.60
		ERNIE-Search_2.4B		19.20	344.06	363.26	-	-	-
	Generative-Reader	EMDR2	(10, 50]	1.76	32.00	33.76	-	52.50	71.40
		YONO_retriever	(50, 100]	1.54	65.00	66.54	-	53.20	71.30
		FiD-base		1.76	65.00	66.76	2.00	48.20	65.00
		FiD-base+KD_DPR		1.76	65.00	66.76	-	49.60	68.80
		YONO_reranker		1.76	65.00	66.76	-	53.20	71.90
		GAR+DPR_generative		2.50	65.00	67.50	1.25*	45.30	-
		RAG-seq		2.50	65.00	67.50	0.80	44.50	56.80
		FiD-large		3.96	65.00	68.96	0.50	51.40	67.60
		FiD-large+KD_DPR		3.96	65.00	68.96	-	53.70	72.10
		RePAQ+FiD_large	(100, 500]	3.32	220.00	223.32	2.30	52.30	67.30
Retriever-Only		RePAQ_base+PQ	(10, 50]	**0.04**	48.00	48.04	100.00	41.20	-
		RePAQ_base	(100, 500]	**0.04**	220.00	220.04	**1400.00**	40.90	-
		RePAQ_base+reranker_base		0.09	220.00	220.09	55.00	45.70	-
		RePAQ_XL		0.24	220.00	220.24	800.00	41.50	-
		RePAQ_XL+reranker_XXL		1.18	220.00	221.18	6.00	47.60	52.10
		DensePhrases		0.88	320.00	320.88	20.60	40.90	50.70
		DenSPI+Sparc	(1000, 2010]	2.69	1547.00	1549.69	2.10	14.50	34.40
		DenSPI		2.69	2000.00	2002.69	2.90	8.10	30.70
Generator-Only		T5-1.1-small+SSM	(0, 10]	0.24	0.00	0.24	7.20*	25.50	-
		T5-base		0.88	0.00	0.88	7.53*	25.90	29.10
		BART-large		1.62	0.00	1.62	5.88*	26.50	26.70
		GAR_generative		1.62	0.00	1.62	2.94*	38.10	62.20
		T5-large		3.08	0.00	3.08	3.85*	28.50	35.90
		T5-1.1-XL+SSM	(10, 50]	12.00	0.00	12.00	-	29.50	45.10
		T5-1.1-XXL+SSM		45.27	0.00	45.27	-	35.20	61.60
		GPT-3	(500, 1000]	700.00	0.00	700.00	-	29.90	71.20

Table 2: The statistical information of Wikipedia corpora used in ODQA models.

Wikipedia Corpus	Split Method	Retrieval Unit	Length of a Unit (tokens)	Number of Units (million)	Encoding Methods	Index Size (GB)	Relatived ODQA models
2016-12-21 dump of English Wikipedia	-	article	-	5.1	TF-IDF	26	DrQA
					BM25	2.4	Skylinebuilder, GAR_extractive
2018-12-20 snapshot of English Wikipedia	BERT's tokenizer	block/ passage	288	13	dense encoding	18	ORQA, REALM
	-	block/ passage	100	21	dense encoding	65	DPR, RocketQA, R2-D2, etc.
	-	phrase	<=20	60000	TF-IDF+ dense encoding	2000	DenSPI
					dense encoding	320	DensePhrases
	generator	QA-pair	-	65	dense encoding	220	RePAQ

build the index for passages on this Wikipedia corpus. The size of the index file is 65GB. Based on this Wikipedia corpus, RePQA further generates 65 million QA pairs and indexes these QA pairs to a 220GB file. Some other methods, e.g. DrQA (Chen et al., 2017) and Skylinebuilder (Wu et al., 2020), encode and build indexes for documents from the 2016-12-21 dump of English Wikipedia which includes 5.1 million articles (Chen et al., 2017; Wu et al., 2020), and the size of this index file is 26GB.

Except for the different choices of the original corpus, there are also different partition and segmentation strategies. For example, ORQA (Lee et al., 2019) and REALM (Guu et al., 2020) segment the corpus documents into 13 million blocks, each of which has 288 tokens. DenseSPI (Seo et al., 2019), Dens+Sparc (Lee et al., 2020) and DensePhrases (Lee et al., 2021b) divide corpus documents into 60 billion phrases, each phrase including 20 tokens. The rest ODQA models segment corpus documents into 21 million passages with a length of 100 tokens, leading to a 65GB index (Karpukhin et al., 2020; Lewis et al., 2021; Izacard and Grave, 2021; Qu et al., 2021).

A comprehensive introduction is illustrated in Table 2. In general, the index size of the corpus is quite large, and the storage of the index is one of the main challenges for ODQA efficiency.

Metrics. There are various metrics to depict efficiency in different dimensions.

In terms of latency, training time (Mao et al., 2021), indexing time (Mao et al., 2021), query time (Yamada et al., 2021) and reasoning time are normally considered. The metrics *Q/s* (questions per second) (Seo et al., 2019) and *FLOPs* (floating point operations) (Guan et al., 2022) are popular in measuring the total processing latency, where *Q/s* is the number of questions one ODQA system can answer per second and *FLOPs* is the number of floating point operations of the model.

In terms of memory, model parameter size, passage corpus size, index size, and training data size are important to influence factors of memory cost (Yamada et al., 2021). We measure the memory consumption for ODQA models using memory units (bytes) of corresponding data (corpus, index, and model) after loading into memory.

In terms of answering quality, EM (Exact Match accuracy) (Chen et al., 2017), F1-score, MRR@k (Mean Reciprocal Rank) (Qu et al., 2021), precision@k, Recall@k and retrieval accuarcy@k (Karpukhin et al., 2020) are normally used to measure the quality of the ODQA models. Specifically, EM is the percentage of questions for which the predicted answers can match any one of the reference answers exactly, after string normalization (Qu et al., 2021). MRR@k is the mean reciprocal of the rank at which the first relevant passage was retrieved (Qu et al., 2021).

In this paper, we adopt metrics on latency, memory, and accuracy to evaluate ODQA models comprehensively. Specifically, we use *Q/s* to measure the processing speed, use *total memory overhead* to evaluate the memory cost, and use *EM* score to estimate the end-to-end answer prediction quality as shown in Table 1.