Q: What's Open-Domain Question Answering? A:

Christopher Sciavolino
Princeton University
cds3@princeton.edu

Abstract

Open-domain question answering is a challenging and fast-moving task in the field of natural language processing. This paper provides an in-depth introduction to the problem space, including relevant background knowledge. I summarize common datasets and methods used to evaluate model effectiveness. With a solid foundation, I continue by outlining numerous state-of-the-art approaches proposed by the NLP community to solve the problem. I conclude with discussion about areas for future work.

1 Introduction and Problem Space

Open-domain question answering (Open-QA) arose from years of work in the field of question answering. Here, I discuss the origins of the task and gradually build up a formal definition of the problem space.

1.1 Question Answering Task

The task of question answering (QA) is the following: given a question q composed of a sequence of word (or sub-word) tokens, predict the answer a. For example, if the question is "What is the smallest penguin species in the world," then the corresponding answer would be "the fairy penguin." Across questions, answers can take various forms including a sequence of tokens (like the previous example), true/false booleans, numbers, or even the lack of an answer (further referred to as the $null$ answer). For most of this survey, I consider *extractive QA*, where the system predicts a substring *within* a greater knowledge source (e.g. Wikipedia) that answers the question.

1.2 Reading Comprehension Task

One way to format a QA task is through reading comprehension (RC), similar to standardized tests like the SAT and the GRE. The model reads a sequence of tokens containing relevant information, called a context paragraph, and predicts the answer to a given question.

More formally, I define the context document $\mathbf{d}$ as a sequence of L_d word tokens: $\{d_i\}_{i=1}^{L_d}$. Given a question of length L_q defined as $\mathbf{q} = \{q_i\}_{i=1}^{L_q}$, the goal is to predict an answer string $\mathbf{a}$ of length L_a defined as $\{a_i\}_{i=1}^{L_a}$. For the problem of extractive QA, $\mathbf{a}$ would be a substring of $\mathbf{d}$, denoted as $\mathbf{a} \subseteq \mathbf{d}$.

1.3 Open-Domain Question Answering Task

Reading comprehension provides a single document to the model for each question-answer pair. Open-QA makes the problem more challenging by *withholding* the context document. Instead, the model has access to a very large set of documents on a wide range of topics. For each question, the model needs to search through the large corpus of information, composed of both relevant and irrelevant documents, to identify the correct answer.

This makes the task significantly more difficult because the model needs to sift through thousands of documents in the knowledge source (commonly Wikipedia), identify the relevant ones, and accurately read the retrieved documents to find the correct answer.

More formally, the model has access to a large collection of N_D documents, $\mathbf{D} = \{D_1, D_2, \ldots, D_{N_D}\}$. For a question-answer pair (q, a), the goal is for the model to take in a question q and search through $\mathbf{D}$ to predict the answer a.

2 Background Knowledge

Many proposed solutions to Open-QA build off of previous milestones from the NLP community. Specifically, one system that dramatically changed the landscape of Open-QA is BERT (Devlin et al., 2019).

Previous embedding methods like word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014) use *static* word embeddings, which have issues with words that vary meaning depending on context like "bank" or "play." To better capture context, many autoregressive (unidirectional) encoders like ELMo (Peters et al., 2018) provide different representations dependent on surrounding context. Although a step in the right direction, autoregressive models still do not utilize bidirectional context, which can capture richer word embeddings.

From these imperfections came Bidirectional Encoder Representations from Transformers, commonly known as BERT (Devlin et al., 2019). BERT uses a fully bidirectional context when encoding words, which is superior to one or two unidirectional contexts. BERT is also based on Transformers (Vaswani et al., 2017), which use a multi-head self-attention mechanism to capture long-term dependencies.

In order to use the fully bidirectional context, BERT considers two different unsupervised training objectives. The first and more impactful task is called Masked Language Modeling (MLM). For this task, BERT replaces 15% of words in the corpus with a special [MASK] token. The goal is for BERT to predict the original word whenever it encounters a [MASK] token.

The second training objective is called Next Sentence Prediction (NSP). BERT samples two sentences from the corpus and predicts whether the second sentence follows the first. For each sentence pair BERT processes, it prepends a special [CLS] token and separates the two sentences using a special [SEP] token.

To obtain a sentence representation, most models use the BERT [CLS] token representation. That is, given a sentence S encoded as $[CLS]\, S\, [SEP]$, the model takes the embedding of the [CLS] token as the embedding of the entire sentence.

After pre-training, BERT can be applied to a downstream task, gradually improving embeddings towards a domain-specific task through an additional training process called *fine-tuning*. In most cases, using BERT to solve a downstream task involves adding a single layer on top of the BERT model. For example, if the task is classification, the top layer takes in an input representation from BERT and make a prediction over the possible labels.

In the case of question answering, BERT takes the sequence of tokens in the context paragraph and predicts two token-level probability distributions: one for whether the word is the start token in the answer span (Pr_s), and one for whether the word is the end token of the answer span (Pr_e). From these distributions, BERT selects the highest probability span with score S_{pred} for indices $start$ and end where,

$$S_{pred} = \underset{start,end}{\operatorname{argmax}}\, Pr_s(w_{start}) \times Pr_e(w_{end})$$

and $start \leq end$. Note that enumerating every span within a large context to calculate a probability score is computationally expensive, so many systems constrain the length of context paragraphs.

3 Datasets

Open-QA is evaluated across many different datasets with various properties. Below I describe the most commonly referenced datasets for the task.

3.1 Stanford Question Answering Dataset (SQuAD)

The Stanford Question Answering Dataset (SQuAD) v1.1 (Rajpurkar et al., 2016) contains around 100k QA examples. Each question-answer pair has an associated context paragraph from Wikipedia, where each answer is a span within the context. The model must predict the answer span within the context paragraph.

The authors created SQuAD by presenting a paragraph from one of 536 Wikipedia articles to crowdsourced workers and asking them to propose a question that can be answered using the provided context. The workers then annotate the correct answer span within the paragraph.

When evaluating a model on SQuAD for Open-QA, most models use "SQuAD-open," where the model evaluates the development set without being provided the context paragraph, as proposed in (Chen et al., 2017). Instead of reading through the context paragraph, the model searches through all of Wikipedia to find the correct answer span.

More recently, Rajpurkar et al. (2018) released SQuAD v2.0, which adds an additional 50k examples to SQuAD and introduces unanswerable questions (i.e. questions with the $null$ answer).

3.2 Natural Questions (NQ)

Natural Questions (NQ) (Kwiatkowski et al., 2019) is a QA dataset with around 325k examples gathered from a pool of Google search queries. Each question has an associated evidence document, a long answer (typically a paragraph), and a short answer (one or more entities within the long answer), with the possibility of there being no answer ($null$).

To generate the dataset, the authors gathered popular search queries matching a set of valid question criteria. The authors then find Wikipedia articles within the top search results for the query and present them to a crowdsourced worker. The annotator identifies a long answer and a short answer, or indicates there is no answer ($null$). For the development and test data, the authors ensure quality by having 5 different workers annotate each question.

Interestingly, the query may not actually be in the form of a question. For example, it could be a "complete the sentence" query, like "the smallest penguin species," or it could be ungrammatical, like "actress in the girl with the dragon tattoo swedish." Some Natural Questions examples have ambiguous acceptance criteria, leading to new datasets and tasks like AmbigQA (Min et al., 2020).

Most Open-QA models evaluate on Natural Question Open (NQ-open) (Lee et al., 2019), which only keeps questions with non-$null$ short answers less than 5 tokens and withholds the given evidence document.

3.3 HotpotQA

HotpotQA (Yang et al., 2018) is a multi-hop question answering dataset with 113k Wikipedia-based QA pairs. Each pair has a question that requires finding and reasoning over multiple supporting documents. This means that rather than only looking at a single document to find the answer, the model must retrieve multiple documents and *combine* the knowledge provided by each to predict the answer for each question.

HotpotQA is sourced using a very specific pipeline. First, the authors construct a Wikipedia graph using hyperlinks as edges between first paragraphs in each article. The authors then manually define a set of valid bridge-entity articles that can be used to create clear multi-hop questions. To generate each example, a crowdsourced worker is provided a randomly sampled bridge-entity article as well as a paragraph linking to it on the Wikipedia graph. The worker then asks a question based on the two paragraphs and labels the correct answer.

Each example has a question-answer pair, two context paragraphs from potentially different documents, and two annotated sentences to support the answer (called supporting facts). Each supporting fact is labelled by the crowdsourced worker within the context paragraphs. Notably, each question-answer pair has exactly two context paragraphs rather than an arbitrary amount of context paragraphs.

4 Evaluation Metrics

In order to make progress on Open-QA, the field needs to define success on a given dataset. Below, I describe different types of evaluation for Open-QA.

Exact Match (EM) The Exact Match (EM) metric directly compares the final predicted answer $\hat{a}$ to the gold answer a and checks word-for-word equality. This helps evaluate the overall, end-to-end correctness of the model.

Note a model can still receive EM credit for predicting the correct answer using invalid or inaccurate supporting evidence, so a high EM score does not *guarantee* perfect model understanding.

F1-Score The F1-score measures the approximate overlap between the predicted answer and the ground truth answer. The score treats both the predicted answer and the gold answer as bag-of-words vectors to calculate *precision* and *recall*. Precision is the number of words in both the predicted answer and the gold answer, divided by the length of the *predicted* answer. Recall is the same numerator divided by the length of the *gold* answer. The F1-score is defined as:

$$F_1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

5 Neural Baseline Models

Prior to 2017, solutions to Open-QA were extraordinarily complex, using tons of hand-crafted features and an amalgam of different subsystems. In this section, I discuss DrQA (Chen et al., 2017), the first successful neural Open-QA system many subsequent models use as their foundation. I then touch on a few additional features that are standard in more recent systems. Finally, I present a stronger, more recent baseline, BERTserini (Yang et al., 2019).

5.1 DrQA: First neural retrieval-reader

Chen et al. (2017) revitalized the Open-QA community with their release of DrQA, the first successful neural approach to the problem. The authors leverage a 2016 dump of Wikipedia as their source of knowledge. DrQA is composed of two components pipelined together: a DocumentRetriever that picks the top-k most relevant documents from Wikipedia and a DocumentReader that finds the best answer within the top-k documents.

The DocumentRetriever component picks the top-k documents using a heuristic search with Term Frequency Inverse Document Frequency (TF-IDF) scoring over bag-of-words vectors. The authors take local word order into account by using efficient bigram hashing from Weinberger et al. (2009) as features for their TF-IDF score.

The DocumentReader component consists of a 2-layer bidirectional LSTM for paragraph representations and a second recurrent neural network for question representations, both using GloVe (Pennington et al., 2014) word embeddings. Consider a single paragraph p_i composed of a series of N tokens $\{p_j\}_{j=1}^{N}$. DrQA computes the paragraph embedding as:

$$\{\mathbf{p}_1, \ldots, \mathbf{p}_N\} = LSTM_{PAR}(\{\tilde{p}_j\}_{j=1}^{N})$$

where $LSTM_{PAR}$ is the 2-layer bidirectional LSTM paragraph encoder and $\tilde{p}_i$ is a concatenation of 300 dimensional GloVe word embeddings with a few handcrafted features.

At inference time, the DocumentReader generates a question embedding by passing the corresponding GloVe word embeddings through a second recurrent neural network. Using both the top-k paragraphs and the question embeddings, the DocumentReader computes probability distributions for the start and end indices of the answer span (likely inspiring BERT). DocumentReader returns the highest scoring $P(start) \times P(end)$ probability where $start \leq end \leq start + 15$, ensuring the answer span is less than 15 tokens.

DrQA contributed a simple, neural, 2-stage pipeline that performs comparably with previous, hand-crafted, complex systems. Still today, 2-stage retriever-reader models are the de facto method for solving Open-QA.

5.2 Natural Extensions

Since DrQA, there are a few additional features common in more recent models: reader document re-ranking, newer document retrieval heuristics, and the replacement of LSTM-based models with BERT.

Reader Document Re-ranking In some models, the reader treats all documents received from the retriever equally; however, some documents are more or less likely to contain the answer according to their TF-IDF score. The reader can incorporate the retriever's scoring into its own overall score, better utilizing the retriever as a guide for prioritizing paragraphs.

Similarly, the reader can re-rank all of the retrieved documents using its own heuristic. Suppose the reader learns that lexical overlap with certain words is more important than TF-IDF ranking. In this scenario, the reader can incorporate this information with the retrieval score to better prioritize the retrieved documents.

Recent Document Retrieval Heuristics In DrQA, the DocumentRetriever uses simple TF-IDF scoring on bag-of-words vectors. More recently, Best Match 25 (BM25) is the default method to determine word-level similarity, outperforming TF-IDF in most systems. Although the two methods are very similar, BM25 weights the TF score asymptotically and uses the document length to better determine how relevant the document is.

Recent Document Reader using BERT Since the introduction of BERT, the Open-QA community shifted from LSTM-based models in favor of BERT models. BERT learns rich contextual information during pre-training and better captures long-term dependencies with the Transformer's superior self-attention mechanism. In almost all systems proposed today, documents are encoded using BERT's [CLS] token representation instead of LSTMs.

5.3 BERTserini: Recent neural baseline

After taking DrQA, updating the DocumentRetriever and DocumentReader modules with more recent components, and changing the name, the model becomes BERTserini (Yang et al., 2019). The system is based on the same retriever-reader model as DrQA, but uses the updated features mentioned above.

Specifically, it uses an Anserini (a BM25 implementation) document retriever with a BERT reader model fine-tuned on SQuAD for QA. For recent work, BERTserini is a much more challenging and accurate baseline compared to DrQA, although the general structure is very similar.

6 Different Directions

From the baselines, the NLP community proposed many different directions to tackle Open-QA. In this section, I discuss many of them, along with examples of models that fall into each category.

6.1 General Pre-trained Models

General pre-trained models like GPT-2 (Radford et al., 2019) and T5 (Raffel et al., 2019) have been applied to Open-QA datasets and performed decently. Even though they aren't specifically created to solve Open-QA, their parameters house a significant amount of information from their respective pre-training tasks. As expected, the more parameters packed into a model, the more information the model is able to retain.

That said, Raffel et al. (2019) demonstrate this approach has diminishing returns. On the Natural Questions dataset, increasing the number of parameters in T5 from 3 billion to 11 billion only improves the test score from 32.1% to 34.5%. That's an increase in 8 billion (!!) parameters for a task improvement of only 2.4%. To put this in perspective, BERT-base has 110M parameters, so adding 8 billion parameters is equivalent to adding 73 BERT-base models!

6.2 Novel Pre-training Tasks

Many recent approaches introduce a novel pre-training task for their model to train on prior to fine-tuning and evaluation. Three notable example of this type of direction are ORQA (Lee et al., 2019), REALM (Guu et al., 2020), and Hard-EM (Min et al., 2019a).

ORQA: Inverse-Cloze Task Pre-training Lee et al. (2019) propose ORQA, an end-to-end retriever-reader model trained jointly on an Inverse Cloze Task (ICT).

The authors segment each Wikipedia article into smaller evidence blocks. ORQA's retriever consists of two BERT models, one for encoding evidence blocks, $BERT_B$, and one for encoding the question, $BERT_Q$. The retrieval score is defined as the inner product between the BERT [CLS] representations of the question and the evidence block. The top-k scoring evidence blocks are passed to the reader.

ORQA's reader uses a third BERT model, $BERT_R$, that identifies the best answer span in each of the top-k evidence blocks using the classic DrQA and BERT-style QA prediction. The reader creates two probability distributions over the evidence for the start and end token indices and defines a span's score as the product between the start and end token probabilities.

ORQA's primary contribution is the ICT pre-training task. In this setup, the retrieval model samples a sentence from a random evidence block and removes it 90% of the time. Given a set of candidate evidence blocks, the goal is to identify the block associated with the sampled sentence. Note that 10% of the time, the sentence is not removed from the evidence block, making the problem trivial. The removal (90%) ensures the model picks up semantic meaning while the remaining examples (10%) allows the model to learn the importance of lexical overlap as a feature.

After the ICT pre-training task, the authors freeze the weights for $BERT_B$ and pre-compute a large index over the evidence blocks using Locality Sensitive Hashing for quick maximum inner product lookups. ORQA fine-tunes on a given dataset, further improving $BERT_Q$ and $BERT_R$ for the downstream task. To encourage more training of $BERT_Q$, they include an additional loss term by looking for exact answer matches within the top-c context documents, where $c \gg k$ by a few orders of magnitude.

ORQA performs well on datasets like Natural Questions, but does not achieve state-of-the-art performance on SQuAD. The authors offer an explanation that SQuAD questions are written given the Wikipedia context, creating a bias where questions have high lexical overlap with the evidence. Additionally, contexts are sampled from only 536 documents, which biases the retrieved articles towards a small subset of Wikipedia. From the results, systems using BM25 and TF-IDF scoring generally *overperform* on datasets like SQuAD due to these biases.

REALM: Retrieval-Augmented Language Model Guu et al. (2020) build on ORQA and propose Retrieval-Augmented Language Model pre-training (REALM). The REALM model is

composed of 2 modules, a Neural Knowledge Retriever and a Knowledge-Augmented Encoder.

The Neural Knowledge Retriever encodes both the document and the question using two BERT models, denoted here as $BERT_{doc}$ and $BERT_q$ respectively. The retriever computes a relevance score using the inner product between the BERT [CLS] representations of the document and the question after being projected into a lower dimensional vector space.

The Knowledge-Augmented Encoder uses a third BERT model for RC, denoted here as $BERT_{read}$, to encode the question and the associated context paragraph concatenated together. In a BERT-like fashion, the Knowledge-Augmented Encoder predicts the start and end indices of the answer span within the question-document pair.

REALM's end-to-end pre-training task begins by sampling a sentence from Wikipedia and masking a token (like MLM). The Neural Knowledge Retriever reads the sentence and retrieves relevant documents. The Knowledge-Augmented Encoder receives the masked sentence along with the top-k retrieved documents and predicts the original masked word. REALM uses ICT pre-training as a warmup prior to REALM pre-training.

One difficulty in this model is computing the probability distribution for the answer given the question, denoted

$$Pr(a|q) = \sum_{d \in \mathcal{D}} Pr(a|q,d)Pr(d|q)$$

where a is the answer string, $\mathcal{D}$ is the set of all available documents, and q is the input question. The authors solve this by using a maximum inner product search (MIPS) index to approximately search for top-k documents efficiently; however, each time the parameters update, the index becomes stale. To fix this, the authors continuously "refresh" the index using the new, updated parameters after a set number of timesteps.

In addition to the ICT warm-up and the modified MLM task, the REALM model includes other small modifications to improve performance. The authors use salient span masking, where the model selects particularly important spans when masking sentences. The retriever also removes the document containing the masked sentence from the retrieved documents to avoid trivial solutions. Finally, some questions require no world knowledge to predict, so the retriever has the option to return an empty $null$ document. After training, the REALM model fine-tunes in a supervised fashion on downstream datasets.

REALM performs exceedingly well, building on ORQA's results on NQ-Open. An interesting point the authors note is that a faster refresh rate of the MIPS index leads to better results, at the expense of more compute power.

Hard-EM: Hard-EM Training Loss Min et al. (2019a) propose Hard-EM, a BM25+BERT model that alters the frequently used maximum marginal likelihood (MML) loss function with a novel parameter update called Hard-EM. Specifically, the authors replace the MML loss,

$$J_{MML}(\theta|x,Z) = -\log \sum_{z_i \in Z} P(z_i|x;\theta)$$

where Z is the candidate set of all possible solutions, with their novel loss function,

$$J_{Hard}(\theta|x,Z) = -\max_{z_i \in Z} \log Pr(z_i|x;\theta)$$

The Hard-EM model pre-computes the top-k scoring paragraphs from Wikipedia using BM25. Using their training objective, the authors fine-tune a BERT model for QA to find the most likely span within the top-k scoring paragraphs.

The authors argue that the MML loss may assign high probabilities to spurious, incorrect answers candidates. Rather than reinforce *all* of the candidates, the Hard-EM loss function only reinforces the top-scoring candidate (by replacing the $\sum$ with a $\max$).

Although simple in concept, the BM25+BERT model trained with the Hard-EM loss performs 2-3% better than the same model trained with the MML loss.

6.3 Multi-hop Question Answering

Multi-hop QA focuses on question-answer pairs requiring reasoning across multiple context documents. Multi-hop QA complicates the retriever because the model may need to retrieve multiple documents about different topics, not necessarily lexically relevant to the question. It also complicates the reader, which needs to understand a *group* of documents instead of reading a single document.

PathRetriever Asai et al. (2020) propose PathRetriever, a retriever-reader model, to reason through multi-hop question-answer pairs. Instead of operating on paragraphs, PathRetriever considers *reasoning paths*, or a series of paragraphs logically connected that together reveal the final answer. The authors consider a Wikipedia graph where each paragraph is a node and hyperlinks between articles represent edges.

During the retrieval stage, the model creates a set of candidate paragraphs C_t composed of the top-F scoring Wikipedia paragraphs using TF-IDF. At each timestep, a recurrent network iteratively selects a paragraph from the candidate set and generates a new candidate set C_{t+1}. The new set contains all of the paragraphs directly connected to the selected paragraph on the Wikipedia graph as well as the top-k scoring paragraphs from the previous candidate set C_t. Each paragraph is encoded using the BERT [CLS] representation of both the question and the paragraph concatenated together. Each paragraph score is defined as the inner product between the recurrent network's hidden state and the candidate representation. A reasoning path can be arbitrarily long and ends when the recurrent network picks a special end-of-evidence [EOE] token. The retriever performs a beam search to output the top B reasoning paths for the reader.

During the reader stage, the reader model has a multi-task objective for (1) reasoning path re-ranking and (2) reading comprehension. Initially, the reader re-ranks the reasoning paths according to the probability it contains the final answer. The reader then selects the highest scoring reasoning path and performs BERT-style QA prediction with the context being the concatenation of all paragraphs in the reasoning path.

To help train both the retriever and the reader, the authors use negative sampling to induce additional loss per example. The retriever also augments the training data by adding high TF-IDF scoring paragraphs within the Wikipedia graph to each gold paragraph, creating an artificial reasoning path.

PathRetriever performs exceedingly well on HotpotQA, but does not achieve significant performance gains on typical Open-QA datasets like NQ-Open, where it falls short of ORQA. The recurrent retrieval mechanism for retrieving evidence documents along with the Wikipedia graph accounts for most of the performance gains on HotpotQA.

6.4 Low-latency Models

Many of the solutions discussed take a long time to search and solve a single query. Most consumer applications desire the lowest latency possible, so they're willing to trade off state-of-the-art accuracy for a shorter response time. Seo et al. (2019) demonstrate it is possible to build accurate Open-QA systems with significantly lower prediction latency.

Dense-Sparse Phrase Index Seo et al. (2019) propose Dense-Sparse Phrase Index (DenSPI) to focus on low-latency Open-QA applications. The key component allowing DenSPI to achieve real-time speeds is a preprocessed index of all *phrases* within the corpus (in this case, Wikipedia).

Both the phrase and question are encoded in 3 parts: a dense vector, a sparse vector, and a scalar. The dense vector captures important semantic information and consists of both the start token representation and the end token representation, each derived using BERT. The question's start token is defined to be the [CLS] token. The sparse representation is used to capture precise lexical information and uses a 2-gram-based TF-IDF model combined with a sparse vector from the surrounding context paragraph. The scalar coherency value is the inner product between the start and end token representations, which ideally captures the inner contents.

The proposed index would need 240 TB (!!) of storage. In order to reduce the memory footprint, the authors use compression techniques like reusing pointers, filtering out irrelevant phrases, and quantization to reduce the storage overhead to 2 TB.

During training, the dense embedding model is trained in a supervised fashion, where the gold paragraph is provided for each question. At inference time, the model searches by either using the dense vector first (dense-first search) or the sparse vector first (sparse-first search). The best performing method uses a combination of both, denoted the *hybrid* approach.

DenSPI achieves performance comparable to baselines like BERTserini on SQuAD-Open; however, DenSPI spends 0.81 seconds per query while BERTserini spends 115 seconds per query. DenSPI also performs 6000x less computation than DrQA and achieves 58x faster end-to-end inference.

6.5 Context Paragraph Representations

A core component of Open-QA is creating a robust, semantic representation for questions and context paragraphs. While sparse embeddings like TF-IDF and BM25 provide a strong baseline, dense representations are space-efficient and useful for inner product search. In this section, I discuss work focusing on building strong dense vector representations for Open-QA.

Dense Paragraph Retriever (DPR) Dense Paragraph Retriever (DPR) (Karpukhin et al., 2020) focuses on improving the retrieval mechanism in Open-QA by only learning dense representations.

DPR generates a dense vector representation for each paragraph in Wikipedia using a BERT [CLS] token representation and indexes them using FAISS (Johnson et al., 2017) offline. FAISS performs efficient similarity search over dense vectors. At inference time, DPR encodes the question using a second BERT model's [CLS] token representation and searches for the nearest top-k passages.

To train DPR, the authors use a combination of positive and negative samples. The positive samples for each question are the gold paragraphs in a given dataset. The negative samples are all other gold paragraphs within the same question's mini-batch as well as a single high-scoring BM25 passage that does not contain the answer string.

When evaluated on retrieval-only tasks, the dense representations from DPR outperform the BM25 sparse representations in top-20 and top-100 accuracy on all datasets except SQuAD. The authors follow the argument of Lee et al. (2019), who describe biases inherit in SQuAD causing TF-IDF and BM25 baselines to overperform on retrieval tasks.

Since both representations encode useful and complementary information, the authors introduce a hybrid representation which re-ranks retrieved passages using a linear combination of BM25 and DPR scores. The authors also train two versions of DPR: one trained using a single dataset and one trained on all of the datasets (except SQuAD).

After adding a BERT QA reader to the trained retriever models, DPR trained with multiple datasets is able to achieve state-of-the-art performance on datasets like Natural Questions.

6.6 Augmented Domain Data

While the task of Open-QA is commonly defined over a set of source documents like Wikipedia articles, there are other sources that can be referenced to help answer questions. In particular, knowledge bases are full on information connecting entities by relations to augment Wikipedia articles. Min et al. (2019b) explore this direction through their proposed model, GraphRetriever.

GraphRetriever Min et al. (2019b) merge the WikiData (Vrandečić and Krötzsch, 2014) knowledge base with a retrieval-reader system to augment the information stored in passages.

GraphRetriever constructs a graph of passages using entity relations defined within the knowledge base. The retriever constructs a candidate set of relevant passages and gradually add more each iteration until reaching a maximum size. The candidate set is initialized using a combination of (1) articles about entities referenced in the question according to an entity linking system (Ferrucci et al., 2010), and (2) the first passages from the top-K_{TFIDF} TF-IDF ranked articles. Each iteration, the model adds passages in two ways. First, the model adds passages from *new* articles using knowledge graph relations from the previous candidate passages. Next, the model adds the top-K_{BM25} scoring passages within the same articles according to BM25.

GraphReader takes the candidate graph constructed by GraphRetriever and encodes each of the knowledge graph relations into a single passage representation. In typical BERT-style, the model uses probability distributions to find start and end tokens of the highest-scoring span.

GraphRetriever + GraphReader yield state-of-the-art results on datasets like Natural Questions, likely due to the fact that the knowledge base identifies important relations previous models cannot. The combination of edges *between* articles and edges *within* articles helps the model navigate Wikipedia effectively and efficiently.

7 Future Directions

Open-QA systems apply numerous different strategies to model the problem, as described in Section 6. In this section, I discuss potential areas for future research.

General pre-trained models: There still remains a very large gap between these models and systems built specifically for Open-QA. A future direction for inquiry is whether progress on Open-QA can be made using general pre-trained models *without* throwing more parameters into the model.

That is, can more general multi-task objective models both generalize well and close the performance gap for Open-QA?

Novel pre-training tasks: I believe REALM's (Guu et al., 2020) pre-training task is particularly successful because it accurately models the task at hand. Specifically, the task is to retrieve information about a question, and predicting retrieved documents to identify masked words within the question seems like a step in the right direction.

A further step forward would be to identify a new pre-training task that better models the goal of retrieving relevant information *missing* from the question in addition to contextual information.

Multi-hop QA: Current models do an excellent job of modeling the idea of reasoning paths or collecting groups of relevant documents; however, I believe the current evaluation methods are lacking. The most frequently used multi-hop QA dataset for Open-QA is HotpotQA, which is constructed in a very specific, artificial way. In addition, context is constrained to two gold paragraphs, which, in my view, does not adequately model the complexity multi-hop QA offers.

An interesting direction would be to construct a new dataset task containing complicated question-answer pairs requiring synthesis of an arbitrary number of documents. Rather than considering 2 documents, the field should instead construct reasoning paths of varying length on the scale of 3-5 paragraphs. This would challenge the RC community to assimilate the context, rather than just search for answers.

Low-latency Models: Returning answers on the order of milliseconds would lead to better user experience and could be integrated in hundred of devices through applications like Siri or Google Home. Although DenSPI reduces the end-to-end inference time significantly, Open-QA systems are far from being commercialized. An analysis of how successful current Open-QA systems are using the compute power of mobile devices would provide an estimate of how far these systems are from on-device prediction.

Memory constraints also pose an issue for on-device Open-QA systems. Specifically, DenSPI requires a 2 TB compressed index, which is far too large for consumer devices today. Is it possible to still answer relevant questions using a subset of Wikipedia rather than the entire, compressed phrase index? Additionally, would techniques like deep compression (Han et al., 2015) work on models like BERT to make large-scale models more mobile friendly?

Context paragraph representations: While BERT's self-attention mechanism is extremely strong, I am not necessarily convinced that it is the *best* way to capture the information stored in a question or in a paragraph. An interesting future research direction would be to investigate how to represent the meaning of a paragraph or a question. One example would be to consider a question as a combination of filters and acceptance criteria. The filters help the retriever identify relevant documents while the acceptance criteria would help the reader model identify the correct answer within the retrieved documents. Can models decouple the representations of (1) what the question is asking and (2) how information in the question helps narrow down where to find the answer?

Augmenting domain data: The use of additional data to augment Wikipedia is a very new and fresh approach for Open-QA. GraphRetriever (Min et al., 2019b) demonstrates strong results by incorporating a knowledge base. I think this direction is underexplored and can be joined with other Open-QA directions. For example, incorporating a novel pre-training task with the Wikipedia/WikiData knowledge source may distill a better understanding of how to effectively use both and can further improve results.

8 Conclusion

Over the last few years, Open-QA methods improved dramatically. Although the task is far from solved, numerous promising directions of future research will continue to drive the field forward.

This survey outlines the problem of Open-QA and describes many popular datasets and evaluation techniques. I discuss important baseline models and explore how proposed systems take divergent strategies to tackle the problem. I finally summarize possible directions for future work along multiple dimensions to push the field incrementally forward.

References

Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. 2020. Learning to retrieve reasoning paths over wikipedia graph

for question answering. In *International Conference on Learning Representations*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Association for Computational Linguistics (ACL)*.

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.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A. Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. 2010. Building watson: An overview of the deepqa project. *AI Magazine*, 31(3):59–79.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training.

Song Han, Huizi Mao, and William J. Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the Conference of Association for Computational Linguistics*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 3111–3119, Red Hook, NY, USA. Curran Associates Inc.

Sewon Min, Danqi Chen, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019a. A discrete hard em approach for weakly supervised question answering. In *EMNLP*.

Sewon Min, Danqi Chen, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019b. Knowledge guided text retrieval and reading for open domain question answering. *arXiv preprint arXiv:1911.03868*.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. Ambigqa: Answering ambiguous open-domain questions.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad.

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

Minjoon Seo, Jinhyuk Lee, Tom Kwiatkowski, Ankur P Parikh, Ali Farhadi, and Hannaneh Hajishirzi. 2019. Real-time open-domain question answering with dense-sparse phrase index. In *ACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledge base. *Communications of the ACM*, 57:78–85.

Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. 2009. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 1113–1120, New York, NY, USA. Association for Computing Machinery.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. End-to-end open-domain question answering with BERTserini. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 72–77, Minneapolis, Minnesota. Association for Computational Linguistics.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.