SN Computer Science (2023) 4:595
https://doi.org/10.1007/s42979-023-02039-x

ORIGINAL RESEARCH

Open-Domain Long-Form Question–Answering Using Transformer-Based Pipeline

Aprameya Dash[1] · **Mohit Awachar**[1] · **Anshul Patel**[1] · **Bhawana Rudra**[1]

Received: 13 March 2023 / Accepted: 6 June 2023

Abstract

For a long time, question–answering has been a crucial part of natural language processing (NLP). This task refers to fetching accurate and complete answers for a question using certain support documents or knowledge sources. In recent years, much work has been done in this field, especially after the introduction of transformer models. However, analysis reveals that the majority of research done in this domain mainly focuses on answering questions curated to have short answers, and fewer works focus on long-form question–answering (LFQA). LFQA systems generate explanatory answers for questions and pose more challenges than the short-form version. This paper investigates the long-form question–answering task by proposing a system in the form of a pipeline consisting of various transformer-based models, enabling the system to give explanatory answers to open-domain long-form questions. The pipeline mainly consists of a retriever module and a generator module. The retriever module retrieves the relevant support documents containing evidence to answer a question from a comprehensive knowledge source. On the other hand, the generator module generates the final answer using the relevant documents retrieved by the retriever module. The Explain Like I'm Five (ELI5) dataset is used to train and evaluate the system, and the final results are documented using proper metrics. The system is implemented in the Python programming language using the PyTorch framework. According to the evaluation, the proposed LFQA pipeline outperforms the existing research works when evaluated on the Knowledge-Intensive Language Tasks (KILT) benchmark and is thus effective in question–answering tasks.

Keywords ELI5 · Long-form question–answering · Natural language processing · Passage retrieval · Transformers

Introduction

Question–answering systems are an integral part of the natural language processing domain. Due to substantial technological improvements and the progress in making the Internet accessible in the nooks and corners of the world, information regarding a large number of topics is now readily available to common people. As a result, question–answering systems are necessary to automatically find and generate answers for the various questions and queries posed by the users. While extracting information using special query languages may be easier, answering a question written in natural language by a user is a very challenging task for a machine. For answering a question, the machine needs to properly understand a question, fetch relevant documents for the given question, and then formulate a response that is not only correct according to the fetched documents but also correct according to the grammar and vocabulary

This article is part of the topical collection "Research Trends in Computational Intelligence" guest edited by Anshul Verma, Pradeepika Verma, Vivek Kumar Singh, and S. Karthikeyan.

✉ Bhawana Rudra
bhawanarudra@nitk.edu.in

Aprameya Dash
aprameyadash2001@gmail.com

Mohit Awachar
mohitawachar2001@gmail.com

Anshul Patel
darshan.patelndps@gmail.com

[1] Department of Information Technology, National Institute of Technology Karnataka, Surathkal, Mangalore, Karnataka 575025, India

Published online: 03 August 2023

of the language. The retrieved answers should be factually, syntactically, and semantically correct.

Over the years, a lot of research [1–4] has been done for creating question–answering systems. Previously, the question–answering systems used to be based on probabilistic or basic vector space models, which do not take the semantic meaning of the questions into consideration. Later, the advancements in word embedding systems allowed the creation of systems that could somewhat improve the understanding of the question and answers. In recent years, the introduction of transformer models opened the door for highly advanced question–answering systems, which can also take advantage of pre-trained transformer models like BERT to get a very high understanding of natural language. However, most of the work done in question–answering systems deals with answering questions that have small answers, mostly consisting of a few words. These systems perform very well in short-form question–answering but do not work well when generating answers for questions that require explanatory, paragraph-length answers.

Recently, the long-form question–answering (LFQA) task has gained attraction in the NLP domain. The LFQA task integrates the retrieval of relevant documents and question–answering. This means that such a system has to fetch support documents relevant to a question from a vast knowledge base consisting of a huge number of support documents related to different fields and topics. Next, the system has to generate a semantically correct and explanatory answer using the relevant documents. Even modern search assistants are only capable of searching relevant web pages for a given question and answering short-form questions but fail to generate rich, abstractive, and explanatory answers by collecting relevant passages from multiple sources across different domains. Thus, this task is quite challenging, and the systems employed for such tasks should be capable of having a rich understanding of natural language.

This paper investigates the LFQA task by proposing a system to answer open-domain long-form questions in the form of a pipeline consisting of different modules that contribute to answering a question. There are two main modules in the proposed pipeline: the retriever module and the generator module. The retriever module consists of a core transformer-based model, which is used to retrieve the most relevant documents from a huge external knowledge source, and a re-ranking model that re-ranks the top-ranked retrieved documents. The second module is the generator module, again a transformer-based generative module that takes the question and the relevant documents fetched by the retriever module and generates an answer for the question. The entire system is trained on the ELI5 dataset [5], a dataset introduced by Facebook consisting of questions with explanatory answers, and Wikipedia is used as the external knowledge source. The proposed system is then evaluated on ELI5's test dataset as well as the KILT benchmark platform [6], and the results are noted and compared with the existing works. In summary, the following are the main contributions of this research work:

1. Creation of a transformer-based retriever module capable of retrieving the relevant documents for an open-domain question from a vast external knowledge source (Wikipedia). The retriever module contains a core retriever model trained to retrieve relevant passages for a question. Next, a re-ranking strategy is integrated into the module to strengthen relevant document/passage retrieval further.
2. Creation of a generator module capable of generating explanatory answers for an open-domain question using the documents retrieved by the retriever module.
3. Extensive experimentation is done by training different transformer architectures as the core retriever model, varying the re-ranking strategy's parameters for relevant document retrieval, and thus, identifying the best configuration for the retriever module. Similarly, different sequence-to-sequence (seq2seq) models are trained for answer generation, and the best configuration is identified.
4. The best-proposed configurations of the LFQA system are also evaluated on the KILT benchmark for the ELI5 dataset. Evaluation reveals that the proposed LFQA systems are very effective and outperform the existing research works in the same field. Further, the system is also evaluated using the BERTScore [7] metric to gain insights into the answer-generation quality from a semantic perspective.

There are a total of five main sections in this paper. "Introduction" briefly introduces the proposed work. "Related work" gives a detailed account of existing datasets and research works in the question–answering domain. "Methodology" describes the methodology proposed in this research work. "Experiments and results" explains the experiments carried out to evaluate the proposed system and then reveals the results of those experiments. Finally, "Conclusion and future work" draws the research work's conclusion and gives suggestions regarding ways to improve the research.

Related Work

Many of the earlier works in the field of question–answering were based on statistical and rule-based techniques. Initially, probabilistic models like Naive Bayes and decision trees were paired up with techniques like bag-of-words, *n*-grams, etc., to create query-answering systems. Further, representation methods like the Term Frequency-Inverse Document

Frequency (TF-IDF) [8–10] were also integrated into the querying systems. Later, methodologies such as the Best Match (BM) models like BM-25 [1] also gathered attraction for use in querying systems. But, then, advancements in computing and the introduction of neural networks influenced the NLP field a lot, and a lot of research has been done to employ deep learning in question–answering systems. Architectures like Recurrent Neural Networks (RNNs) and their more advanced versions, Gated Recurrent Unit (GRU), and Long Short-Term Memory (LSTM) [11] were also used a lot due to their inherent capabilities to handle long sequences [12, 13]. Later, the introduction of transformer models, especially pre-trained ones, like Bidirectional Encoder Representations from Transformers (BERT) [14, 15] influenced this field a lot, and these transformer models became very useful in the most recent question–answering systems due to their capability to understand a sentence.

Datasets used for training question–answering systems are generally of two types, extractive or abstractive. Extractive datasets are those in which the answer is a phrase directly taken from the provided context itself, whereas abstractive datasets deal with more general and natural answers. The SQUAD [16] dataset and the NewsQA [17] dataset are prominent examples of extractive datasets. SQUAD is a reading comprehension dataset with more than 100,000 questions created using a set of Wikipedia articles. The answer for each question is a phrase directly taken from the Wikipedia document/passage associated with the question. The NewsQA extractive dataset, on the other hand, contains about 100,000 question–answer pairs created using a set of over 10,000 articles. On the other hand, some prominent examples of abstractive datasets include NarrativeQA [18], a book and movie summaries dataset, and CoQA [19], a dialogue dataset. The NarrativeQA dataset is a more complex dataset in which a reader has to answer questions by reading entire books or movie scripts and thoroughly understanding the context and narrative. The CoQA dataset contains about 127K questions with answers and is built using a huge number of conversational texts from different domains. All these datasets consist of answers which range from a few words to a few sentences and are thus suited for short-form question–answering systems. The ELI5 dataset [5], introduced by Facebook, is one of the very few datasets suited for long-form question–answering containing questions with an average word count of 42.6 and answers having an average of 130.6 words.

Over the years, many research works have focused on short-form question–answering (QA). Many pre-trained standard transformer models, as well as seq2seq models, which are generalized to perform well in large number of NLP tasks, also achieve great results when evaluated on short-form question–answering datasets such as SQUAD. Chen et al. [20] proposed a two-stage model consisting of a document retriever and a machine comprehension model to achieve competitive results in multiple short-form QA datasets. Tang et al. [21] explored the power of pre-training by combining 77 datasets to create a huge dataset MVPCorpus and pre-trained a text generation model to achieve excellent performance in QA tasks. Bhojanapalli et al. [22] improved the standard transformers by eliminating redundancy in pairwise dot product-based attention and achieved competitive results in many datasets, including SQUAD. Riabi et al. [23] proposed a method to improve cross-lingual QA. Huang et al. [24] proposed new ways to improve transformer models by suggesting a relative position embedding technique and received improved performance on QA datasets like SQuAD1.1. However, these works often perform poorly when employed in LFQA tasks.

Nitish et al. [25] investigated the LFQA task by extracting long procedural answers from long PDFs using transformer-based models like BERT, trained on a self-made dataset ProLongQA. The paper also implements the long-form question–answering system using different transformer-based models such as BERT and its variants like RoBerta, DistilBert, Albert, etc. Results displayed that BERT and RoBerta worked the best, achieving accuracies of 87.2% and 86.4%, respectively. In Fan et al. [5], the ELI5 dataset is introduced. This dataset is created by selecting questions and answers from a subreddit *Explain Like I'm Five (ELI5)*. The answers in this dataset are explanatory, long, and generally written in simple and plain language. The paper also examines various methods like TF-IDF, BidAF, etc., for relevant document retrieval and extractive models. Next, the paper investigates the use of language models and Seq2Seq models for answer generation and concludes that a multi-tasking Seq2Seq model works the best. The paper also analyses different metrics that can be used for evaluating LFQA models. Bui et al. [4] proposed a self-made dataset, FitQA, containing question–answer pairs related to nutrition. The paper then uses various state-of-the-art models to train on this dataset and provides a detailed comparison between the models. The paper also discovers that the same models which work very well on datasets like SQUAD perform poorly when trained on a dataset containing questions with longer answers. The model also tries techniques like curriculum learning to improve the performance of the models. Butler et al. [26] focused on creating an accurate, open-domain, deployable question–answering system that uses different modules for zero-shot classification, relevant document retrieval, and answer generation to generate answers for open-domain questions. The model is also built to be sufficiently efficient for deployment purposes.

For a completely automated and independent LFQA system, retrieving passages containing evidence to answer a question and serve as support documents is also quite important. This passage retrieval is mainly done using a

vast knowledge source such as Wikipedia. Guu et al. [27] introduced an intuitive retrieval technique for LFQA tasks called REALM or Retrieval-Augmented Language Model which involves training a retrieval model on unsupervised text using a performance-based signal. The model is then employed in the open domain question–answering task, and analysis reveals that the model outperformed many previous methods by a significant margin. Karpukhin et al. [28] introduced another retrieval technique, DPR or Dense Passage retrieval, which aims at using a dual-encoder architecture to increase the dot product of the representations of query and relevant documents. The DPR model learns embeddings using a dual-encoder framework from a small number of question–answer pairs and can defeat the existing methods by a good margin. Lee et al. [29] introduced Open Retrieval Question–Answering system (ORQA), an open-domain question–answering system in which no separate retrieval system is used, and the retriever and the reader are jointly trained using only question–answer pairs, after pre-training the retriever with an Inverse Cloze Task. The proposed system is then evaluated on five different question–answering datasets, and the system outperformed traditional IR systems like BM-25 in most cases.

Lewis et al. [30] explored Retrieval-Augmented Generation (RAG) models for language generation by combining non-parametric and parametric memory. The work proposed and compared two different formulations of the RAG model, fine-tuned the proposed models on a wide variety of NLP tasks, and set state-of-the-art performance in many question–answering tasks. Krishna et al. [2] proposed a question–answering system that uses contrastive REALM (C-REALM) to retrieve relevant documents from a knowledge source and a routing transformer-based model to generate the final answer using the retrieved documents, trained on the ELI5 dataset. The paper also identifies various shortcomings of the ELI5 dataset and lists challenges that must be overcome to improve LFQA systems. The paper also explains that the metrics mentioned in the ELI5 paper are not informative enough, and better metrics are needed to create better LFQA systems. The paper also suggests ways that can be used to mitigate these problems. In Nakano et al. [3], a GPT-3 model is trained on the ELI5 dataset to generate explanatory answers, and the relevant documents are retrieved using the MS Bing API. This way, the paper fine-tunes a language model to use a web-browsing environment. The paper further uses reinforcement learning, imitation learning, and other similar methods to optimize answer quality. The model is built in such a way that it generates answers supported with references to make human evaluation easier. Su et al. [31] investigated the generation of accurate answers to open-domain long-form questions by proposing the Read Before Generate (RBG) model, which models machine reading and generation of answers jointly. The proposed model is an end-to-end framework in which answer-related information is used to strengthen the generator model and emphasize correct facts.

Analysis of existing research works reveals that, even though a lot of work has been done in the general question–answering domain, there are still very few works in the domain of long-form question–answering. Further, while many datasets are present in the question–answering domain like SQUAD, NewsQA, etc., all these datasets are suited for short-form question–answering. ELI5 is one of the few public datasets suitable for LFQA. Similarly, there are fewer works in the field of LFQA due to the inherent complexity of the task. Thus, there is ample opportunity for more research in the LFQA domain.

Methodology

This section provides a detailed description of the proposed question–answering pipeline responsible for providing an explanatory answer for a question using retrieved documents containing evidence for the question.

Dataset Description

The ELI5 dataset used in this paper is suited for long-form question answering. The original dataset, introduced in [5], contains about 272 K question–answer pairs. Out of this, about 237 K question–answer pairs belong to the training dataset, 10 K to the validation dataset, and about 25 K to the testing dataset. The average question length is 42.6 words, whereas the average answer is 130.6 words long. The dataset has been constructed by selecting a set of questions and answers from a subreddit Explain Like I'm Five (ELI5). This subreddit encourages users to post questions on any topic, and answers are usually written in a very simple and natural language, so that even users without pre-existing knowledge of the topic can understand the answers. A vast majority of the questions are 'Why'and 'How' type questions. About 63% of the questions have more than one answer, and the score of each answer (number of up-votes in Reddit) is also present in the dataset. The most important fields present in a sample of the dataset are the question itself, a list of answers, and scores assigned to each answer.

The external knowledge source used in this paper for retrieving support documents for a question is KILT Wikipedia. This knowledge source is vast and holds information related to various fields, making it suitable for open-domain question–answering systems. KILT Wikipedia is a standardized Wikipedia snapshot created by the KILT Benchmark, containing about 5.9 million Wikipedia articles. The knowledge source is divided into snippets of length 100, and each snippet is considered a separate document for retrieval tasks.

This creates a knowledge source containing about 30 million passages or documents in total.

LFQA Pipeline Overview

The LFQA pipeline proposed in this paper consists of two main modules, a retriever module and a generator module. The retriever module takes the question as input and retrieves the support documents relevant to the question from the knowledge source. The retriever module ranks the documents present in the knowledge source. Based on this ranking, the K most relevant documents for the question are selected and passed on to the generator module. These retrieved documents will serve as the context for generating the answer to the question. The generator module takes the question and the context (consisting of documents received from the retriever) and generates the answer. This way, the retriever and the generator module together empower the pipeline to answer open-domain questions. Figure 1 shows the high-level architecture of the proposed LFQA pipeline.

Retriever Module

The retriever module is responsible for retrieving the support documents related to a particular question from a large knowledge source.

Data Pre-processing

The first step in document retrieval is the data pre-processing step. In this step, the question–answer pairs are extracted from the ELI5 training dataset and processed, so that the input can be directly fed to the core retrieval model, a transformer-based encoder model. In the ELI5 dataset, each question may have multiple answers. Therefore, to keep the training process highly robust, one of its answers is selected randomly for each question in every epoch. Then, a sufficiently sized, randomly chosen span of this selected answer is considered as the relevant passage for the question. This ensures that each question is paired up with a slightly different relevant passage in every epoch, which can improve the generality of the model. Next, each question–passage pair is lightly cleaned to remove links, extra whitespaces, etc., and then passed through the tokenizer corresponding to the core retrieval model. Each question is converted to 64-length vectors, whereas each passage is converted to 128-length vectors. Tokenization results in the following vectors for each question/passage:

- Input Token IDs: A vector of the token IDs of the tokens present in the input sequence. These token IDs are assigned based on the embeddings of the tokenizer.
- Attention Mask: A vector to differentiate between the tokens corresponding to the actual sequence and the padding tokens. The value will be 1 for a token belonging to the actual text and 0 for a padding token.

After tokenization, the tokenized question–passage pairs are sent to the core retriever model. As the relevant passage for a question changes in every epoch, tokenization is also performed after every epoch to the altered question–passage pair.

Core Retriever Model and Training

In its core, the retriever module contains a transformer-based model. Each training step feeds a batch of tokenized questions and corresponding relevant passages into the model. Each question and passage pair separately passes through the core transformer-based encoder, which returns a 768-length vector for each question/passage. Gradient check-pointing is performed, so that large batch sizes can be used even in the presence of limited GPU memory. The embedding for each question is passed through a dense layer consisting of 128 neurons, which reduces the question vector to a 128-length projection. Similarly, the embedding of each passage is passed through a separate dense layer with 128 neurons to convert the passage to a 128-length projection. Next, a custom loss function is employed to train the model in such a way that the dot product of the projection of a question with its corresponding passage is higher than the dot product of the projection of that question and any other passage present in the batch. In this way, the model learns to project the question and any passage in such a way that the dot product

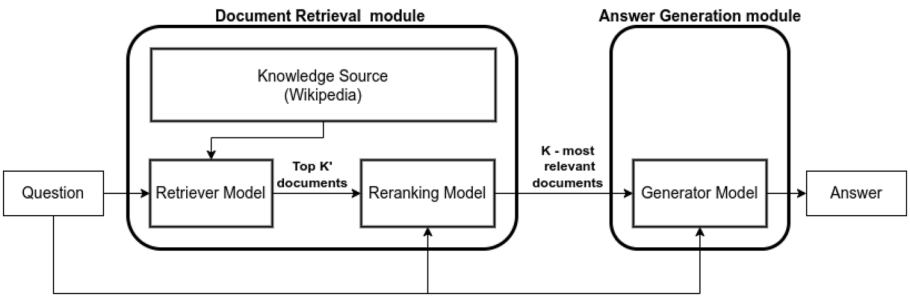

Fig. 1 High-level architecture of LFQA pipeline

of the question and the passage will be higher if the passage is relevant to the question. Figure 2 shows the architecture of the core retrieval model of the LFQA pipeline.

Retrieving Top K' Passages

Once the model is trained, it can be used for ranking purposes. First, the model can be separated into two sub-models: a question encoder that converts a question into 128-length projection and a passage encoder that converts a passage into 128-length projection. Next, each passage from the knowledge source is passed through the passage encoder, and the 128-length encoding of each passage is stored. Once this is done, the ranked list of passages (documents) for a given question can be found. For this, first, the question is passed through the question encoder, and the 128-length projection of the question is computed. Next, the dot product of the question projection vector and each of the pre-stored passage projection vectors is computed. Passages for which the dot product is larger will be given a higher ranking. As the number of passages in the knowledge source is too large, thus, the ranking based on the dot product is done using the Faiss library [32], which is built and optimized for this purpose. This way, a ranked list of passages (documents) is retrieved. Top K' passages from this list are selected for each question and passed on for re-ranking. Figure 3 shows the process for retrieving and ranking passages for a question from a Knowledge Source.

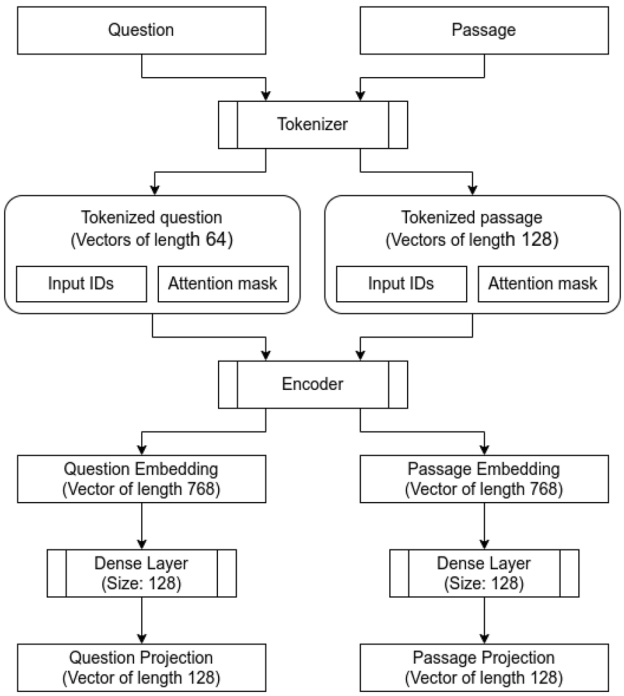

Fig. 2 Architecture of retrieval model

Re-ranking Top K' Passages

The top K' passages retrieved using the retriever model are re-ranked using the zero-shot re-ranking technique introduced by Sachan et al. [33]. The method involves re-scoring the top K' passages to determine the probability of a question generation conditioned on each retrieved passage using a Pre-trained Language Model (PLM). This method is a zero-shot re-ranking strategy; thus, it does not require any domain-specific training and can be used directly for any type of passage re-ranking. Given a question q and an ordered list of retrieved passages $Z = \{z_1, z_2, ..., z_{k'}\}$, the new relevance score for a passage z_i is calculated by $p(z_i|q)$ which can be estimated by computing the quantity $p(q|z_i)$. A pre-trained language model is used to calculate $log\ p(q|z_i)$, which can be defined as the mean log-likelihood of each of the tokens of the question, q, conditioned on the passage z_i. This is done for each retrieved passage of a question using a PLM in a zero-shot manner by adding an appropriate natural language instruction after the passage and passing this new passage to the PLM. After calculating $log\ p(q|z_i)$ for each passage for a given question, the passages are re-sorted according to the new relevance score, and the top K passages are passed on to the generation module. The top K passages

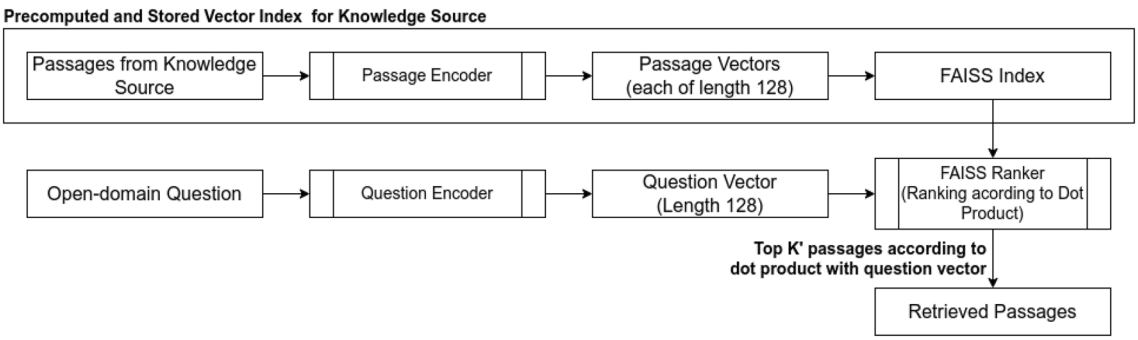

Fig. 3 Process of retrieving passages for a question from a knowledge source

for each question of the training dataset are then retrieved and stored, so that the generator module can directly use these passages. Figure 4 shows the re-ranking process of top K' passages retrieved from the retriever model.

Answer-Generator Module

The generator module takes the top *K* documents (or passages) retrieved by the retriever module for a given question and generates the answer to that question using the retrieved passages as context. The answering is done by a fine-tuned sequence-to-sequence (seq2seq) model.

Data Pre-processing

The top *K* relevant passages for each question are retrieved for each question using the best-performing retriever module. Next, the top *K* passages for each question are concatenated using a special token acting as the separator to create the context for that question. Next up, the question and its context are concatenated using proper prefixes like *question:* and *context:* to create the input sentence, whereas the best-rated answer of the question is chosen as the target sequence. In the case of multiple answers to a question, answers whose score is higher than a threshold are also chosen as the target sequences and paired up with the input sequence. This results in a list of input and target sequence pairs. Next, each pair of input and target sequence is tokenized to get the following vectors:

- Input Token IDs: A vector of the token IDs of the tokens present in the input sequence, according to the vocabulary of the tokenizer. These vectors are padded at the end.
- Attention Mask: A vector to differentiate between the tokens corresponding to the actual sequence and the padding tokens.
- Labels: A vector of the token IDs of the tokens present in the target sequence, with padding at the end. Padding tokens are replaced by −100, so that these tokens will be ignored (masked).

The length of the input IDs and the attention mask for an input–target sequence pair is 1024, whereas the length of the labels is kept between 320 and 400 depending on the seq2seq model used.

Core Generator Model and Training

A pre-trained seq2seq model is used as the core generator model. To improve its performance in the long-form question–answering domain, the model is fine-tuned using the question-answer pairs of the ELI5 dataset. To make the seq2seq model fit for generation purposes, a sequence-to-sequence language modeling head is placed on top of the seq2seq model. The language modeling head helps in text generation as well as in training the model for text generation tasks. The tokenized vectors for each question–answer pair are then grouped into batches of appropriate sizes, and these batches are used to fine-tune the core seq2seq model.

To reduce the training time, data parallelism is used, which replicates the model on multiple devices at once, and then, a portion of an input batch is handled by each replica. The gradients of each replica are then accumulated into the main model. This way, each batch is processed in parallel, leading to a shorter training time. The model is trained to minimize the language modeling loss, through which it learns to generate an appropriate answer given a question and its context as input in a proper format. Further, as the pre-trained seq2seq models are quite large and the batch size is small, thus, instead of updating the weights after each batch, the gradients are accumulated over multiple batches, and then, the weights are updated, leading to more stable training. Also, experimentation of hyperparameters is done depending on the seq2seq model used to enhance the training process.

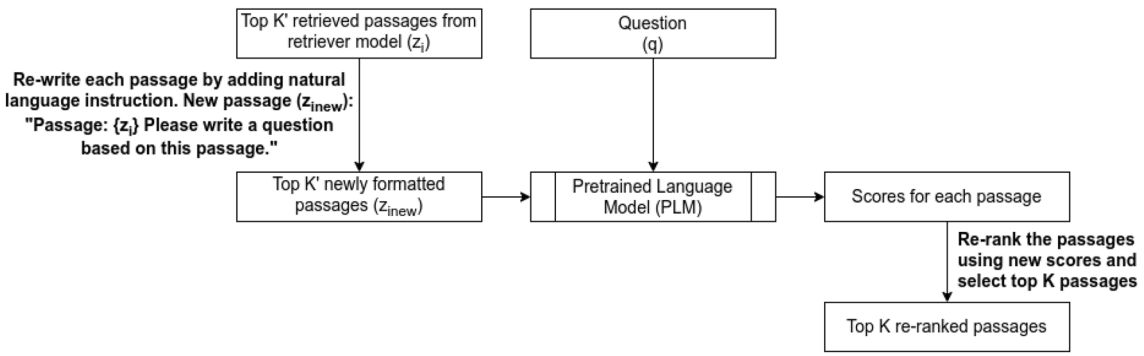

Fig. 4 Process of re-ranking passages

Post-training Utilization

Once the training of the generator model is completed, the pipeline becomes complete and is ready to use. The retriever module fetches the relevant documents for a question, and the generator module generates the answer for the question using the retrieved documents (context). Figure 5 shows the process of answer generation given a question and a list of relevant passages containing evidence for answering that question. Next up, extensive evaluation of both the retriever and the generator module is done on the test partition of the ELI5 dataset as well as the KILT benchmark platform, and the performance of the proposed models is compared with the previous works. The evaluation process and results, as well as the inferences drawn from the extensive evaluation, are discussed in detail in the next section.

Experiments and Results

This section provides a detailed description of the various experiments performed to evaluate the proposed question–answering system, the evaluation results of those experiments, and the inferences drawn from the evaluation. As the retriever and generator modules are two main components of the proposed pipeline, thus, the evaluation results of the two modules are detailed separately.

Evaluation of Retriever Module

Many experiments are conducted to evaluate the retriever module and find the best configuration. The following different variations of the core retrieval model are trained and evaluated:

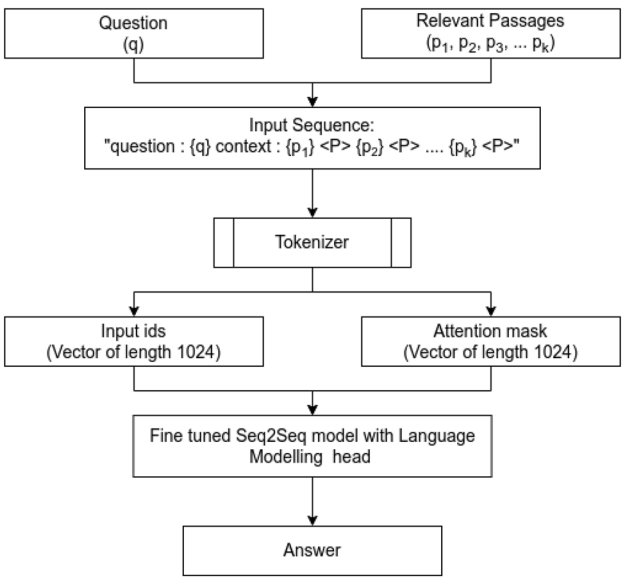

Fig. 5 Process of answer generation

- BERT [15] (bert-base-uncased): BERT stands for Bidirectional Encoder Representations from Transformers. It is a standard transformer model pre-trained on a huge text corpus with two objectives: Next sentence prediction (NSP) and Masked language modeling (MLM). The *bert-base-uncased* variant of the BERT model from HuggingFace [34] is used as the encoder for the core retrieval model.
- MPNet [35] (multi-qa-mpnet-base-dot-v1): MPNet is a powerful pre-training strategy devised to take advantage of two popular models: BERT and XLNet, simultaneously avoiding their limitations. The *multi-qa-mpnet-base-dot-v1* model, available on HuggingFace, is an MPNet-based sentence transformer model trained on multiple question–answering datasets, which can be used to map paragraphs into fixed-length vectors by preserving their semantic meanings.
- DistilBert [36] (multi-qa-distilbert-dot-v1): DistilBert is the distilled version of BERT; it is smaller and faster and uses knowledge distillation. The *multi-qa-distilbert-dot-v1* model, available on HuggingFace, is a DistilBert-based sentence transformer model trained on multiple question–answering datasets, which can be used to generate fixed-length embeddings containing semantic meaning for a paragraph.

For each configuration, the retriever model is trained for ten epochs on the ELI5 training dataset using AdamW optimizer in a GPU-accelerated environment. In the case of the BERT-based retriever model, different variations of this model are trained using batch sizes of 512, 1024, 2048, and 4096. The batch size which produced the best results is then used as the batch size for the other two models. Further, for the re-ranking model of the retrieval module, the pre-trained language model, *T0_3B* [37], is used, which is a transformer-based PLM trained with an objective similar to Masked language modeling.

Each fine-tuned retriever model is then evaluated using the test partition of the ELI5 dataset. All the answers present in the test dataset are collected to create a knowledge source for evaluation. The testing dataset contains about 25K questions, and in total, there are about 75K passages (answers) constituting the knowledge source. For each question, the retriever module should be able to rank that question's answers as high as possible, whereas the answers to other questions should be at lower positions in the ranking. For evaluation, the following metrics are considered:

- Recall@K: It gives the proportion of relevant documents present in the top K retrieved documents for a given

query. The Recall@K value for all queries is averaged to get the final value.

- R-Precision: If R is the number of relevant documents for a given query, then this metric gives the proportion of top R documents which are relevant. The R-precision value for all queries is averaged to get the final value.
- Questions with at least one relevant retrieved passage @K (Q1RRP@K): It gives the proportion of questions for which at least one relevant passage is present in the top K retrieved passages.

The results of the evaluation of BERT-based retriever models are shown in Table 1. Among the different variations of the BERT-based model, the model trained on a batch size of 2048 performs the best. A larger batch size provides a better result as it increases the generalization of the model. Therefore, the other models (mpnet and distilbert) are also trained on a batch size of 2048. Evaluation of these models using the same strategy, as shown in Table 2, reveals that the best-performing model is the one in which *multi-qa-distilbert-dot-v1* is used as the encoder. As this model performed the best, thus, re-ranking is performed on the top 50 retrievals of the *multi-qa-distilbert-dot-v1*-based model, and the new results are recorded. It can be noted that re-ranking improves the performance of the best-performing model, as is evident from Table 2.

To make the task even more challenging, the knowledge source explicitly created for testing using the answers present in the testing dataset is extended by adding all answers present in the training dataset, bringing the number of passages to about 880K. Table 3 displays the evaluation results on the extended knowledge source. Again the model in which *multi-qa-distilbert-dot-v1* is used as the encoder performs the best, and re-ranking the top 50 retrievals enhances the performance of this model. The benefits of re-ranking are quite noticeable in this case, as re-ranking increases the values of the metrics by a considerable amount.

Next, the best-performing model, the one which uses *multi-qa-distilbert-dot-v1* as encoder, is evaluated on the

Table 1 Comparison of BERT-based retriever models trained with different batch sizes across different metrics (in %)

Metric	Bert-base-uncased (BS:512)	Bert-base-uncased (BS:1024)	Bert-base-uncased (BS:2048)	Bert-base-uncased (BS:4096)
Recall@5	44.36	47.26	**48.86**	48.19
Recall@10	54.93	58.18	**59.65**	59.01
Recall@20	64.74	67.81	**69.30**	68.32
Recall@50	75.69	78.05	**78.98**	78.21
Q1RRP@5	66.15	69.25	**70.69**	70.09
Q1RRP@10	75.16	77.68	**78.94**	78.30
Q1RRP@20	82.07	84.15	**85.18**	84.45
Q1RRP@50	88.67	90.07	**90.68**	90.13
R-Precision	31.67	34.43	**36.07**	35.08

The results generated by the best-performing model for each metric are highlighted in bold

Table 2 Comparison of best BERT-based retriever model with all other transformer models across different metrics (in %)

Metric	Bert-base-uncased (BS: 2048)	Multi-qa-mpnet-base-dot-v1 (BS: 2048)	Multi-qa-distilbert-dot-v1 (BS: 2048)	Multi-qa-distilbert-dot-v1+ re-ranking (BS: 2048)
Recall@5	48.86	46.49	52.99	**56.09**
Recall@10	59.65	57.41	63.50	**66.43**
Recall@20	69.30	67.14	72.41	**74.82**
Recall@50	78.98	77.81	81.89	**81.89**
Q1RRP@5	70.69	68.41	74.67	**77.27**
Q1RRP@10	78.94	77.18	82.05	**84.11**
Q1RRP@20	85.18	83.90	87.37	**88.89**
Q1RRP@50	90.68	90.06	92.39	**92.39**
R-Precision	36.07	33.26	40.07	**43.65**

The results generated by the best-performing model for each metric are highlighted in bold

Table 3 Comparison of some of the best-performing retriever models on the extended knowledge source across different metrics (in %)

Metric	Bert-base-uncased (BS: 2048)	Bert-base-uncased (BS: 4096)	Multi-qa-distilbert-dot-v1 (BS: 2048)	Multi-qa-distilbert-dot-v1+re-ranking (BS: 2048)
Recall@5	20.69	20.36	24.64	**29.62**
Recall@10	27.95	27.24	32.18	**37.01**
Recall@20	36.21	35.52	40.78	**44.72**
Recall@50	48.82	48.14	53.36	**53.36**
Q1RRP@5	36.52	36.05	42.05	**49.07**
Q1RRP@10	46.39	45.43	51.77	**57.34**
Q1RRP@20	56.22	55.59	61.32	**65.42**
Q1RRP@50	69.24	68.46	73.28	**73.28**
R-Precision	13.63	13.20	16.64	**21.21**

The results generated by the best-performing model for each metric are highlighted in bold

already established KILT benchmark. The KILT benchmark is a unified benchmark platform that allows evaluation for multiple datasets covering a wide range of tasks. The benchmark provides a standardized Wikipedia Knowledge source and a curated test dataset for ELI5 containing 600 samples, whose answers are hidden from the public. For creating a submission to the KILT platform, the best-performing retrieval model, the one based on *multi-qa-distilbert-dot-v1* is used to retrieve top K' passages for every question from the Wikipedia knowledge source. Next, these passages are re-ranked using the re-ranking model, and the top K passages ($K = 10$) of the re-ranked list are submitted for each question. The value of K' is varied ($K' = 50, 100, 200$) to evaluate the performance at different values of K'. The KILT platform evaluates retrieval systems using two metrics: R-precision and Recall@5. The results are shown in Table 4.

From Table 4, it can be concluded that re-ranking improves the performance a lot after the initial passage retrieval. The configuration in which $K' = 50$, i.e., the top 50 passages from the retrieval stage are re-ranked, gives the best performance. Also, as seen from the table, the proposed models give better or competitive results when compared to most of the existing works.

Based on the evaluation of the retrieval module, it is evident that using *multi-qa-distilbert-dot-v1* as the encoder of the retriever model gives the best results. This can be due to the fact that this encoder is already pre-trained on a large number of question–answer datasets. Further, training the model on large batch sizes (e.g., 2048) improves the performance as larger batch size results in more generalization. Furthermore, even though re-ranking is comparatively expensive, re-ranking the retrieved passages using a PLM also improves the performance of the retrieval module. The performance of the proposed retrieval module is comparable to or better than many of the existing works and is, thus, effective at performing passage retrieval for questions from a large knowledge source.

Evaluation of Answer-Generator Module

The experiments conducted to evaluate the answer-generator module are described here. The seq2seq model with language modeling head is the core component of the generator model. Thus, each of the following three pre-trained seq2seq models is used as the core model for experimentation:

Table 4 Evaluation of retriever module on KILT benchmark and comparison with other related works (bottom 4 models are proposed models)

Model	R-precision	Recall@5
RAG	11.0	22.9
RBG	10.83	**27.25**
BART + DPR	10.7	26.9
RT + REALM	6.7	15.7
RT + c-REALM	10.7	24.6
Multi-qa-distilbert-dot-v1 retriever	8.33	18.47
Multi-qa-distilbert-dot-v1 retriever + re-ranking with $K' = 50$	**11.33**	23.27
Multi-qa-distilbert-dot-v1 retriever + re-ranking with $K' = 100$	11.33	22.69
Multi-qa-distilbert-dot-v1 retriever + re-ranking with $K' = 200$	11.16	22.94

The results generated by the best-performing model for each metric are highlighted in bold

- bart-large [38]: BART (Bidirectional Auto-Regressive Transformers) is a standard encoder–decoder transformer architecture that can be used for seq2seq tasks such as translation. It consists of a decoder like GPT and an encoder like BERT, thus, taking advantage of both.
- T5-large [39]: T5 (Text-to-Text-Transfer-Transformer) is a seq2seq that achieves excellent performance in multiple NLP tasks by assessing every NLP task as a text-to-text task; that is, both the input and the target sequences are text sequences. The model is pre-trained on C4 (Colossal Clean Crawled Corpus) using different supervised and unsupervised objectives.
- switch-base-8 [40]: Switch transformer is a new family of transformers similar to the T5 model, with the most crucial difference being that sparse MLP layers are used to replace the feed-forward layers of the T5 model. This allows faster training and good generalization in multiple NLP tasks.

Each of the above pre-trained models is trained using the training dataset according to the proposed methodology in a GPU-accelerated environment. The *bart-large* and *T5-large* models are trained for two epochs, and the *switch-base-8* model is trained for three epochs. The *bart-large* and *switch-base-8* models are trained with a learning rate of 2e−4 and AdamW optimizer. In contrast, the *T5-large* model is trained using a learning rate of 4e−4 and Adafactor optimizer. The relevant passages for each question are retrieved using the best-performing retriever model, which uses the *multi-qa-distilbert-dot-v1* as the encoder, and the top K ($K = 10$) passages for each question are retrieved for this purpose (after re-ranking). Once the models are trained, the models are stored for inference and evaluation.

The answer generated for a sample question by the *bart-large*-based generator module is shown below:

Question: How did people clean their teeth before the invention of the toothbrush?

Answer: Before the invention of the toothbrush, people used to chew on twigs and leaves to clean their teeth. The bristles of a toothbrush are the same as the bristles on a twig, so it is not that hard to clean your teeth with twigs or leaves. It is also worth noting that toothbrushes have been around for a very long time, and people have been using them for thousands of years. Before the modern day, people did not brush their teeth as much as they do now, so they did not have to worry about brushing their teeth like we do now. They did not need a brush, and they just used twigs.

Next, the answer-generation capability of the three models is evaluated on the KILT benchmark platform. The test set available on the KILT benchmark contains 600 questions. The top $K(K = 10)$ passages for these questions are extracted using the retriever module, which uses *multi-qa-distilbert-dot-v1* as the encoder. Next, each of the three trained seq2seq models is used to generate answers for the questions using the corresponding relevant passages as context. The answers are then submitted to the KILT benchmark in the prescribed format. The benchmark evaluates the system answers through the following metrics:

- ROUGE-L: This metric measures the longest matching sequence of words between the system answer and the reference answer.
- F1: This metric calculates the harmonic mean of the recall and precision for a given reference and system answer pair, thus giving a single measure by combining recall and precision.
- KILT-RL & KILT-F1: These two metrics are calculated by combining retrieval results with ROUGE-L and F1 score, respectively. For each data point in the test set, if the R-precision is 1, then the benchmark awards ROUGE-L and F1 points to KILT-RL and KILT-F1, respectively. This way, points are awarded to the system answer only when the relevant passages for the question are located at the top of the retrieved passages.

Table 5 shows the performance of the three trained answer-generator models and compares these results with that of existing works on the KILT benchmark. The bottom three entries of Table 5 refer to the proposed models, whereas the rest belong to other related works. It can be inferred from the comparison that all three proposed models perform better than all the previous works in all metrics except ROUGE-L, in which they are slightly behind one of the works. Further, among the proposed models, it can be inferred that the question–answer-generation pipeline, which uses a trained *bart-large* as the core seq2seq model, performs the best. Thus, the

Table 5 Evaluation of answer-generator module on KILT benchmark and comparison with other related works (bottom 3 models are proposed models, DBR: *multi-qa-distilbert-dot-v1*-based retriever; RR: re-ranking)

Model	ROUGE-L	F1	KILT-RL	KILT-F1
T5-base	19.08	16.10	0.00	0.00
BART	20.55	19.23	0.00	0.00
RAG	14.05	14.51	1.69	1.79
RBG	**24.53**	27.13	2.62	3.00
BART + DPR	17.41	17.88	1.90	2.01
RT + c-REALM	23.19	22.88	2.36	2.34
DBR + RR + bart-large	24.29	**27.70**	**2.70**	**3.12**
DBR + RR + T5-large	24.39	27.38	2.68	3.12
DBR + RR + switch-base-8	23.87	27.46	2.64	3.05

The results generated by the best-performing model for each metric are highlighted in bold

Table 6 BERTScore values (precision, recall, and F1) achieved by each of the proposed models when evaluated on the ELI5 test dataset (DBR: *multi-qa-distilbert-dot-v1* based retriever; RR: Re-ranking)

Model	Precision	Recall	F1
DBR + RR + bart-large	0.839	0.837	0.838
DBR + RR + T5-large	0.843	0.837	0.840
DBR + RR + switch-base-8	0.844	0.836	0.840

proposed models are able to outperform the existing works in almost all metrics and are, thus, capable of performing the question–answering in a better way.

Although the KILT benchmark only uses metrics like ROUGE-L and F1, which work on the syntactic overlap (exact matching) between the reference answer and the system answer, these metrics are incapable of performing a comparison by taking the semantic similarity into account. For this reason, the proposed models are also evaluated using BERTScore, a metric that performs comparison by understanding what is generated (system answer) and what is supposed to be generated (reference answer). This metric works by taking advantage of the pre-trained contextual embeddings of a transformer model, such as BERT or RoBERTa, and then using cosine similarity to match words in the system and reference answers. As a result, this metric gives a more comprehensive evaluation of the generated answers by taking semantic meaning into account and is proven to be more similar to human judgment. Table 6 shows the BERTScore values for the three proposed models when evaluated on the ELI5 test dataset.

BERTScore comprises three metrics: precision, recall, and F1, and each of these metrics is a value between 0 and 1. Table 6 shows that the three proposed models have similar performance, and all the values are much closer to 1, which implies that the proposed models performed well and generated answers which are very close to the reference answers.

Computational Efficiency

To measure the computational efficiency of the proposed system, the average duration taken by each module to process a given question is calculated. According to the experimentation, the core retriever model takes about 0.002 s/question using a single Tesla V100 GPU. The re-ranking model takes about 0.53 s/question with a 4-GPU setup, 0.94 s/question with a 2-GPU setup, and about 2 s/question with a single GPU setup. The final answer generator module takes about 3 s/question to generate an answer using a single GPU setup, once the relevant passages are available. These durations can vary according to the type and number of GPUs used and can decrease further with more powerful GPU setups.

Inferences and Important Takeaways

From the extensive evaluation of both the retriever module and the generator module, it can be inferred that the proposed question–answering pipeline outperforms the existing research works in answer generation in almost all metrics on the KILT benchmark. The pipeline configuration that uses *multi-qa-distilbert-dot-v1* as the encoder in the retriever model, a PLM, *T0_3B*, for passage re-ranking and a trained *bart-large* as answer-generator model performs the best in most metrics, with the other proposed configurations following closely behind. The accurate retrieval capabilities of the retrieval module can be accredited to the contrastive learning-based training objective of the *multi-qa-distilbert-dot-v1*-based retriever module, using large batch size, that enabled the model to project relevant passages closer to the projection of their corresponding question. The re-ranking model further improved the retrieval results by re-ordering the top retrieved passages for a question based on the probability that the question can be generated from each of the passages, calculated using a powerful PLM in an unsupervised manner. Finally, the accurate answer-generation capabilities of the answer-generator model can be accredited to the trained seq2seq model, for instance, *bart-large*, trained to take the question and relevant passages as input and generate an output answer by minimizing the language modeling loss. Evaluation using the BERTScore metric, which takes semantic meaning into account, also reveals that the answer generation of proposed models is quite close to the reference answer. However, the pipeline is quite resource intensive as it consists of multiple language models, which are computationally expensive and thus require a proper GPU environment for good performance and latency, as is evident from the computational efficiency measurement results. Further, in some cases, when the evidence for answering the question is not present in the retrieved support documents, it is observed that the answer generator model will attempt to get information from the data it is pre-trained and generate some text which may not contain the actual answer. Apart from this, the proposed question–answering pipeline can be used to answer questions from different domains automatically and correctly, and further research can be done to improve the pipeline.

Conclusion and Future Work

Systems for answering questions are a crucial and essential component of the natural language processing (NLP) field. The long-form question–answering (LFQA) task

has grown in popularity recently in the NLP field. The LFQA task incorporates both question–answering and document retrieval. In this paper, a pipeline for the LFQA task, consisting of a retriever module and a generator module, is constructed using various transformer-based models. The retriever module is responsible for fetching support documents relevant to a question. In contrast, the generator module is the one that generates the answer to the question using the retrieved documents. Evaluation of the retriever module reveals that using encoders pre-trained on question–answering datasets, using large batch sizes, and using re-ranking strategies are effective in boosting the performance of retrieval models. The best-performing module, which uses a fine-tuned *multi-qa-distilbert-dot-v1* as encoder and a PLM, *T0_3B* for re-ranking achieves 11.33 in R-Precision and 23.27 in Recall@5 on the KILT benchmarking platform, which is comparable to or better than many of the existing works and is thus effective at performing passage retrieval for open-domain questions. Next, the evaluation of the answer generation module shows that a properly fine-tuned sequence-to-sequence model can generate correct and meaningful answers for a question using the retrieved documents as context. The best-performing model, which is a fine-tuned *bart-large* model, achieves 24.29 in ROUGE-L, 27.70 in F1, 2.70 in KILT-RL, and 3.12 in KILT-F1, outperforming the existing research works. Evaluation using the BERTScore metric also confirms that the generated answers are semantically close to ground truth. Further improvement can be made by finding ways to maintain the performance using computationally lighter models and improving the core retriever model and generator module, such that the generated answers are more grounded in the retrieved passages.

Data Availability The Dataset and Knowledge Source used and analyzed in the current study, i.e., ELI5 [5] and KILT Wikipedia [6] respectively, are publicly available and can be used with proper citations.

Declarations

Conflict of Interest The authors declare that they have no competing interests.

References

1. Chen W, Chang M-W, Schlinger E, Wang W, Cohen WW. Open question answering over tables and text. 2020. arXiv preprint arXiv:2010.10439.
2. Krishna K, Roy A, Iyyer M, Hurdles to progress in long-form question answering. 2021. arXiv preprint arXiv:2103.06332.
3. Nakano R, Hilton J, Balaji S, Wu J, Ouyang L, Kim C, Hesse C, Jain S, Kosaraju V, Saunders W et al. Webgpt: browser-assisted question-answering with human feedback. 2021. arXiv preprint arXiv:2112.09332.
4. Bui M-Q, Tran V, Nguyen H-T, Le Nguyen M, How state-of-the-art models can deal with long-form question answering. In: Proceedings of the 34th Pacific Asia conference on language, information and computation, 2020; pp. 375–382.
5. Fan A, Jernite Y, Perez E, Grangier D, Weston J, Auli M, Eli5: long form question answering. 2019. arXiv preprint arXiv:1907.09190.
6. Petroni F, Piktus A, Fan A, Lewis P, Yazdani M, De Cao N, Thorne J, Jernite Y, Karpukhin V, Maillard J et al. Kilt: a benchmark for knowledge intensive language tasks. 2020. arXiv preprint arXiv:2009.02252.
7. Zhang T, Kishore V, Wu F, Weinberger KQ, Artzi Y. Bertscore: evaluating text generation with bert. 2019. arXiv preprint arXiv:1904.09675.
8. Bahri S, Sumpeno S, Nugroho SMS. An information retrieval approach to finding similar questions in question–answering of Indonesian government e-procurement services using tf* idf and lsi model. In: 2018 10th international conference on information technology and electrical engineering (ICITEE). New York: IEEE; 2018. p. 626–631 .
9. Huang X, Zhang Y, Wei B, Yao L, A question–answering system over traditional Chinese medicine. In: 2015 IEEE international conference on bioinformatics and biomedicine (BIBM). New York: IEEE; 2015. p. 1737–1739.
10. Mohapatra SK, Upadhyay A. Using TF-IDF on Kisan call centre dataset for obtaining query answers. In: 2018 international conference on communication, computing and internet of things (IC3IoT). New York: IEEE; 2018. p. 479–482.
11. Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. 1997;9(8):1735–80.
12. Karimi E, Majidi B, Manzuri MT. Relevant question answering in community based networks using deep lSTM neural networks. In: 2019 7th Iranian joint congress on fuzzy and intelligent systems (CFIS). New York: IEEE; 2019. p. 1–5.
13. Chen L, Zeng G, Zhang Q, Chen X, Wu D, Question answering over knowledgebase with attention-based lstm networks and knowledge embeddings. In: 2017 IEEE 16th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC). New York: IEEE; 2017. p. 243–246.
14. Luo D, Su J, Yu S, A bert-based approach with relation-aware attention for knowledge base question answering. In: 2020 International Joint Conference on Neural Networks (IJCNN). New York: IEEE; 2020. p. 1–8.
15. Devlin J, Chang M-W, Lee K, Toutanova K. Bert: pre-training of deep bidirectional transformers for language understanding. 2018. arXiv preprint arXiv:1810.04805.
16. Rajpurkar P, Zhang J, Lopyrev K, Liang P. Squad: 100,000+ questions for machine comprehension of text. 2016. arXiv preprint arXiv:1606.05250.
17. Trischler A, Wang T, Yuan X, Harris J, Sordoni A, Bachman P, Suleman K, Newsqa: a machine comprehension dataset. 2016. arXiv preprint arXiv:1611.09830.
18. Kočiskỳ T, Schwarz J, Blunsom P, Dyer C, Hermann KM, Melis G, Grefenstette E. The narrative QA reading comprehension challenge. Trans Assoc Comput Linguist. 2018;6:317–28.
19. Reddy S, Chen D, Manning CD. COQA: a conversational question answering challenge. Trans Assoc Comput Linguist. 2019;7:249–66.
20. Chen D, Fisch A, Weston J, Bordes A. Reading Wikipedia to answer open-domain questions. 2017. arXiv preprint arXiv:1704.00051.
21. Tang T, Li J, Zhao WX, Wen J-R. MVP: multi-task supervised pre-training for natural language generation. 2022. arXiv preprint arXiv:2206.12131.

22. Bhojanapalli S, Chakrabarti A, Veit A, Lukasik M, Jain H, Liu F, Chang Y-W, Kumar S. Leveraging redundancy in attention with reuse transformers. 2021. arXiv preprint arXiv:2110.06821.
23. Riabi A, Scialom T, Keraron R, Sagot B, Seddah D, Staiano J. Synthetic data augmentation for zero-shot cross-lingual question answering. 2020. arXiv preprint arXiv:2010.12643.
24. Huang Z, Liang D, Xu P, Xiang B. Improve transformer models with better relative position embeddings. 2020. arXiv preprint arXiv:2009.13658.
25. Nitish S, Darsini R, Shashank G, Tejas V, Arya A. Bidirectional encoder representation from transformers (bert) variants for procedural long-form answer extraction. In: 2022 12th international conference on cloud computing, data science & engineering (confluence). New York: IEEE; 2022. p. 71–76.
26. Butler R, Duggirala VD, Banaei-Kashani F. ILFQA: a platform for efficient and accurate long-form question answering. In: Proceedings of the fifteenth ACM international conference on web search and data mining; 2022. p. 1565–1568.
27. Guu K, Lee K, Tung Z, Pasupat P, Chang M, Realm: retrieval-augmented language model pre-training. 2020. arXiv preprint arXiv:2002.08909.
28. Karpukhin V, Oğuz B, Min S, Lewis P, Wu L, Edunov S, Chen D, Yih W-T. Dense passage retrieval for open-domain question answering. 2020. arXiv preprint arXiv:2004.04906.
29. Lee K, Chang M-W, Toutanova K. Latent retrieval for weakly supervised open domain question answering. 2019. arXiv preprint arXiv:1906.00300.
30. Lewis P, Perez E, Piktus A, Petroni F, Karpukhin V, Goyal N, Küttler H, Lewis M, Yih W-T, Rocktäschel T. Retrieval-augmented generation for knowledge-intensive NLP tasks. Adv Neural Inf Process Syst. 2020;33:9459–74.
31. Su D, Li X, Zhang J, Shang L, Jiang X, Liu Q, Fung P. Read before generate! Faithful long form question answering with machine reading. 2022. arXiv preprint arXiv:2203.00343.
32. Johnson J, Douze M, Jégou H. Billion-scale similarity search with GPUS. IEEE Trans Big Data. 2019;7(3):535–47.
33. Sachan DS, Lewis M, Joshi M, Aghajanyan A, Yih W-T, Pineau J, Zettlemoyer L. Improving passage retrieval with zero-shot question generation. 2022. arXiv preprint arXiv:2204.07496.
34. Wolf T, Debut L, Sanh V, Chaumond J, Delangue C, Moi A, Cistac P, Rault T, Louf R, Funtowicz M. Transformers: state-of-the-art natural language processing. In: Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations; 2020. p. 38–45.
35. Song K, Tan X, Qin T, Lu J, Liu T-Y. Mpnet: masked and permuted pre-training for language understanding. Adv Neural Inf Process Syst. 2020;33:16857–67.
36. Sanh V, Debut L, Chaumond J, Wolf T, Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. 2019. arXiv preprint arXiv:1910.01108.
37. Sanh V, Webson A, Raffel C, Bach SH, Sutawika L, Alyafeai Z, Chaffin A, Stiegler A, Scao TL, Raja A et al. Multitask prompted training enables zero-shot task generalization. 2021. arXiv preprint arXiv:2110.08207.
38. Lewis M, Liu Y, Goyal N, Ghazvininejad M, Mohamed A, Levy O, Stoyanov V, Zettlemoyer L, Bart: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. 2019. arXiv preprint arXiv:1910.13461.
39. Raffel C, Shazeer N, Roberts A, Lee K, Narang S, Matena M, Zhou Y, Li W, Liu PJ. Exploring the limits of transfer learning with a unified text-to-text transformer. J Mach Learn Res. 2020;21(140):1–67.
40. Fedus W, Zoph B, Shazeer N. Switch transformers: scaling to trillion parameter models with simple and efficient sparsity. 2021.