applied sciences

MDPI

Article

You Don't Need Labeled Data for Open-Book Question Answering

Sia Gholami * and Mehdi Noori

Amazon Web Services, San Francisco, CA 94111, USA; nmehdi@amazon.com
* Correspondence: gholami@amazon.com

Abstract: Open-book question answering is a subset of question answering (QA) tasks where the system aims to find answers in a given set of documents (open-book) and common knowledge about a topic. This article proposes a solution for answering natural language questions from a corpus of Amazon Web Services (AWS) technical documents with no domain-specific labeled data (zero-shot). These questions have a yes–no–none answer and a text answer which can be short (a few words) or long (a few sentences). We present a two-step, retriever–extractor architecture in which a retriever finds the right documents and an extractor finds the answers in the retrieved documents. To test our solution, we are introducing a new dataset for open-book QA based on real customer questions on AWS technical documentation. In this paper, we conducted experiments on several information retrieval systems and extractive language models, attempting to find the yes–no–none answers and text answers in the same pass. Our custom-built extractor model is created from a pretrained language model and fine-tuned on the the Stanford Question Answering Dataset—SQuAD and Natural Questions datasets. We were able to achieve 42% F1 and 39% exact match score (EM) end-to-end with no domain-specific training.

Keywords: AWS technical documentation; extractive language models; information retrieval systems; zero-shot open-book question answering

Citation: Gholami, S.; Noori, M. You Don't Need Labeled Data for Open-Book Question Answering. *Appl. Sci.* **2022**, *12*, 111. https://doi.org/10.3390/app12010111

Academic Editor: Valentino Santucci

Received: 15 November 2021
Accepted: 19 December 2021
Published: 23 December 2021

1. Introduction

Question answering (QA) has been a major area of research in artificial intelligence and machine learning since the early days of computer science [1–4]. The need for a performant open-book QA solution was exacerbated by rapid growth in available information in niche domains, the growing number of users accessing this information, and the expanding need for more efficient operations. QA systems are especially useful when a user searches for specific information and does not have the time—or simply does not want—to peruse all available documentation related to their search to solve the problem at hand.

In this article, open-book QA is defined as the task whereby a system (such as a computer software) answers natural language questions from a set of available documents (open-book). These questions can have yes–no–none answers, short answers, long answers, or any combination of the above. In this work, we did not train the system on our domain-specific documents or questions and answers, a technique called zero-shot learning [5]. The system should be able to perform with a variety of document types and questions and answers without training. We defined this approach as "zero-shot open-book QA". The proposed solution is tested on the AWS Documentation dataset. As the models within this solution are not trained on the dataset, the solution can be used in other similar domains, such as finance and law.

Software technical documentation is a critical piece of the software development life-cycle process. Finding the correct answers for one's questions can be a tedious and time-consuming process. Currently, software developers, technical writers, and marketers are required to spend substantial time writing documents such as technology briefs, web content, white papers, blogs, and reference guides. Meanwhile, software developers and solution architects have to spend a lot of time searching for information they need to solve their problems. Our approach to QA aims to expedite this process.

Our work's key contributions are:

1. Introducing a new dataset in open-book QA.
2. Proposing a two-module architecture to find answers without context.
3. Experimenting on ready-to-use information-retrieval systems.
4. Inferring text and yes–no–none answers in a single forward pass once we find the right document.

The rest of the paper is structured as follows: First, related previous work is summarized. Then, the dataset is described. Next, details on implementing the zero-shot open-book QA pipeline are provided. In addition, the experiments are explained, and finally, the results, along with limitations and next steps, are presented.

2. Related Work

There are a number of datasets in the literature for natural language QA [6–15], along with several solutions to answer these questions [16–26]. In this paper, we propose an approach that differs from the previous body of work as we do not receive the context but assume that the answer lies in a set of readily available documents (open-book), and we were not allowed to train our models on the given questions or set of documents (zero-shot). Our proposed solution attempts to answer questions from a set of documents with no prior training or fine-tuning (zero-shot open-book question answering).

2.1. QA Approaches

The natural language QA solutions take a question along with a block of text as context and attempt to find the correct answer to the question within the context. Open-book QA solutions take a question along with a set of documents that may contain the answer, then the solution attempts to find the answer to the original question within the available set of documents. Open-book QA solutions have been explored by several research teams including Banerjee et al. [27], who performs QA using fine-tuned extractive language models, and the work of Yasunaga et al. [28], who performs QA using GNNs.

2.2. Our Inspirations

The inspiration for our two-step extractor–retriever approach came from the work of Banerjee et al. [27] which consisted of six main modules: Hypothesis Generation, Open-Book Knowledge Extraction, Abductive Information Retrieval, Information Gain-based Reranking, Passage Selection, and Question Answering. This solution tries to answer a question from a set of available documents along with a database of common knowledge. Alternatively, earlier works [29] created a logical form from the natural language text and then used formal reasoning to answer questions. In our solution, we were able to combine and condense these steps into two modules without the need of a common knowledge database and a logical form to answer open-book questions.

For our extractor, our approach is inspired by extractive question-answering solutions presented by Devlin et al. [18]. Devlin et al. showed that a bidirectional transformer-based [17] model pretrained on unlabeled text can be fine-tuned with only one extra output layer to achieve good performance on variety of NLP tasks, including language inference, text classification, sentiment analysis, and question answering.

3. Data

Real-world open-book QA use cases require significant amounts of time, human effort, and cost to access or generate domain-specific labeled data. For our solution, we intentionally did not use any domain-specific labeled data and conducted experiments on popular QA datasets and pretrained models. We used feedback from customers to generate a set of 100 questions as the test dataset and used QA datasets, explained in Sections 3.2 and 3.3, for training.

3.1. AWS Documentation Dataset

Herein, we present the AWS documentation corpus (https://github.com/siagholami/aws-documentation, accessed on 10 November 2021), an open-book QA dataset, which contains 25,175 documents along with 100 matched questions and answers. These questions are inspired by the author's interactions with real AWS customers and the questions they asked about AWS services. The data was anonymized and aggregated. All questions in the dataset have a valid, factual, and unambiguous answer within the accompanying documents; we deliberately avoided questions that are ambiguous, incomprehensible, opinion-seeking, or not clearly a request for factual information.

3.1.1. Questions and Answers

There are two types of answers: text and yes–no–none answers. Text answers range from a few words to a full paragraph sourced from a continuous block of words in a document or from different locations within the same document. Every question in the dataset has a matched text answer. Yes–no–none (YNN) answers can be yes, no, or none, depending on the type of question. For example, the question: "Can I stop a DB instance that has a read replica?" has a clear yes or no answer, but the question "What is the maximum number of rows in a dataset in Amazon Forecast?" is not a yes or no question, and therefore obtains "None" as the YNN answer. Here, 23 questions have "Yes" YNN answers, 10 questions have "No" YNN answers, and 67 questions have "None" YNN answers. Table 1 shows a few examples from the dataset.

Table 1. Sample questions from AWS Documentation dataset.

Question	Text Answer	YNN Answer
What is the maximum number of rows in a dataset in Amazon Forecast?	1 billion	None
Can I stop a DB instance that has a read replica?	You can't stop a DB instance that has a read replica.	No
What is the size of a null attribute in DynamoDB?	length of attribute name + 1 byte	None
In Amazon RDS, what is the storage for all DB instances?	100 TB	None
Can I run codePipeline in a VPC?	AWS CodePipeline now supports Amazon Virtual Private Cloud (Amazon VPC) endpoints powered by AWS PrivateLink	Yes
Is AWS IoT Greengrass HIPAA compliant?	Third-party auditors assess the security and compliance of AWS IoT Greengrass as part of multiple AWS compliance programs. These include SOC PCI FedRAMP HIPAA and others.	Yes

3.1.2. Annotation Process

Answers are provided by authors and reviewed by a pool of five experts. We divided the annotation tasks into four steps, where all four steps are completed by one annotator and verified by one expert. The guidelines given to annotators are stated below:

1. Question validation: Annotators determine whether a question is valid or invalid. A valid question is a fact-seeking question that has a clear and unambiguous answer. An invalid question is incomprehensible, vague, opinion-seeking, or not plainly a query for factual information. Annotators must determine this by the question's content only.
2. Document selection: Annotators select the right source document from the AWS Documentation corpus that contains the answer. Every valid question in the dataset must have an accompanying source document.

3. Text answer: If a question is valid and has a valid source document, annotators select the correct answer from the source document. Every question must have a clear and unambiguous text answer. For example, for the question "What is the maximum number of rows in a dataset in Amazon Forecast?", there is one and only one answer which is "1 billion".
4. YNN answer: In the final step, annotators flag the answer's YNN answer as "Yes", "No", or "None", depending on the question and the source document. Every question must have a clear and unambiguous YNN answer. For example, for the question "Can I run codePipeline in a VPC?", there is one and only one answer, which is "Yes".

3.2. SQuAD Datasets

The Stanford Question Answering Dataset (SQuAD) (https://rajpurkar.github.io/SQuAD-explorer/, accessed on 10 November 2021) is a reading comprehension dataset [6], including questions created by crowdworkers on Wikipedia articles. The answers to these questions is a segment of text from reading passages, or the question might be unanswerable. SQuAD1.1 comprises 100,000 question–answer pairs on more than 500 articles. SQuAD2.0 adds 50,000 unanswerable questions written adversarially by crowdworkers to look similar to answerable ones.

3.3. Natural Questions Dataset

The Natural Questions (NQ) dataset (https://ai.google.com/research/NaturalQuestions, accessed on 10 November 2021) includes 400,000 questions and answers created on Wikipedia articles [7]. The corpus contains questions from real users with answers that can be long (a few sentences), short (a few words) if present on the page, or null if no long or short answer is present.

4. Approach

Our approach consists of two high-level modules: a retriever and an extractor. Given a question, the retriever tries to find a set of documents that may contain the answer; then, from these documents, the extractor tries to find the answer. Figure 1 illustrates a high-level workflow of the solution, and Table 2 shows an example of the question, retrieved documents, and extracted answers using the solution.

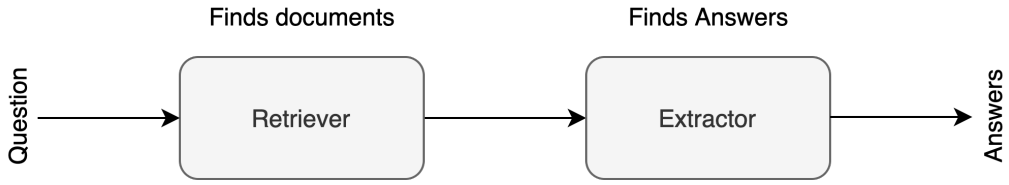

Figure 1. High-level workflow of the solution in one pass.

Table 2. Our solution with an example.

Question:	**What Are the Amazon RDS Storage Types?**
Retriever set of documents:	CHAP_Storage.txt, CHAP_Limits.txt, CHAP_BestPractices.txt
Extractor Text Answer:	General Purpose, SSD, Provisioned IOPS, Magnetic
Extractor Yes/No Answer:	None

4.1. Retrievers

Given a question with no context, our approach relies on the retriever to find the right documents that contains the answer. The need for a retriever stems from the fact that our extractors are fairly large models and it is time- and cost-prohibitive for the extractor to go through all available documents. For example, in our AWS Documentation dataset from

Section 3.1, it will take hours for a single compute instance to run an extractor through all available documents. We conducted experiments with simple information-retrieval systems with a keyword search along with deep semantic search models to list relevant documents for a question. We used precision at K (*P@K*) metric to evaluate our retrievers. Precision at K is the proportion of retrieved items in the top-k set that are relevant:

$$P@K = \frac{\text{number of retrieved documents that are relevant}}{\text{total number of retrieved documents}}$$

4.1.1. Whoosh

Whoosh (https://whoosh.readthedocs.io/, accessed on 10 November 2021) is a fast, pure Python search engine library. The primary design impetus of Whoosh is that it is pure Python and can be used anywhere Python is running, as no compiler or Java is required. It lets you index free-form or structured text and find matching documents based on simple or complex search criteria. For information retrieval, Whoosh uses the Okapi BM25F [30] ranking function, which is a bag-of-words retrieval function that ranks a set of documents based on the query terms appearing in each document. Since Whoosh is easy to implement, easy to maintain and easy to customize, a lot of teams are using it for their information-retrieval use cases.

4.1.2. Amazon Kendra

Amazon Kendra (https://aws.amazon.com/kendra/, accessed on 10 November 2021) is a semantic search and question answering service provided by AWS for enterprise customers. Kendra allows customers to power natural language-based searches on their own AWS data by using a deep learning-based semantic search model to return a ranked list of relevant documents. At the core of its search engine, Amazon Kendra understands natural language questions and returns the most relevant documents. Under the hood, it has a deep learning semantic search model to return a ranked list of relevant documents.

- Document ranking (DR)—Similar to any traditional search engine, Kendra returns a ranked list of relevant documents based on the input query to fulfill their information needs. A deep semantic search model is used to understand natural language questions in addition to the keyword search.
- Passage ranking (PR)—Kendra ranks the passages and tries to find the top relevant passages with a deep reading comprehension model.
- FAQ matching (FAQM)—If there exists frequently answered questions and their corresponding answers, Kendra will automatically match a newcoming query with FAQs and extract the answer if a strong match is found.

4.2. Extractors

Given a question with no context, the retriever finds a set of documents. Then, the output of the retriever will pass on to the extractor to find the correct answer for the original question. We built our extractor from a base model, created from different variations of BERT [18] language models, and added three fine-tuning layers to extract yes–no–none answers and text answers. Our extractor attempts to find YNN answers and text answers in the same pass.

We used F1 and exact match (EM) metrics to evaluate our extractors. EM is a binary metric determining whether the model's prediction was exactly correct (i.e., the same words and in the same order). F1 metric is more relaxed and treats answers as a bag-of-words and calculates the harmonic mean of precision and recall.

$$\text{F1} = \frac{2}{\text{recall}^{-1} + \text{precision}^{-1}}$$

4.2.1. Extractor Model Data Processing

For preprocessing, we followed the work of Devlin et al. [18]. We first tokenized every document using a 30,522 wordpiece vocabulary, then constructed the examples by combining a "[CLS]" token, tokenized question, a "[SEP]" token, tokens from the document, and a final "[SEP]" token. The examples had a maximum sequence length of 512 tokens. For the documents longer than 512 tokens, we used a sliding window over the document tokens with a stride of 128 tokens. The YNN answers are simply transformed to three classes (i.e., yes, no, and none). For the text answers, we computed start and end token indices to represent the target answer span. In the AWS Documentation dataset (Section 3.1), some of the text answers are not from a continuous block of text and therefore there will be more than one text answer spans (i.e., several start and end indices) for a single answer.

4.2.2. Extractor Model

Our extractor model is created with a base model and three fine-tune layers. We used a YNN layer, start indices layer, and end indices layer.

For our base model, we compared BERT (tiny, base, large) [18] along with RoBERTa [31], AlBERT [32], and distillBERT [33]. We implemented the same strategy as the original papers to fine-tune these models. We also used the same hyperparameters as the original papers: L is the number of transformer blocks (layers), H is the hidden size, and A is the number of self-attention heads.

The YNN fine-tune layer takes the pooled output from the base BERT model and feeds it into a three-node dense layer with a Softmax activation to obtain the YNN answer. Furthermore, our model takes the sequence output from the base BERT model and feeds it into two sets of dense layers with sigmoid as activation. The first layer aims to find the start indices of the answer sequences, and the second layer aims to detect the end indices of the answer sequences. Figure 2 illustrates the extractor model architecture.

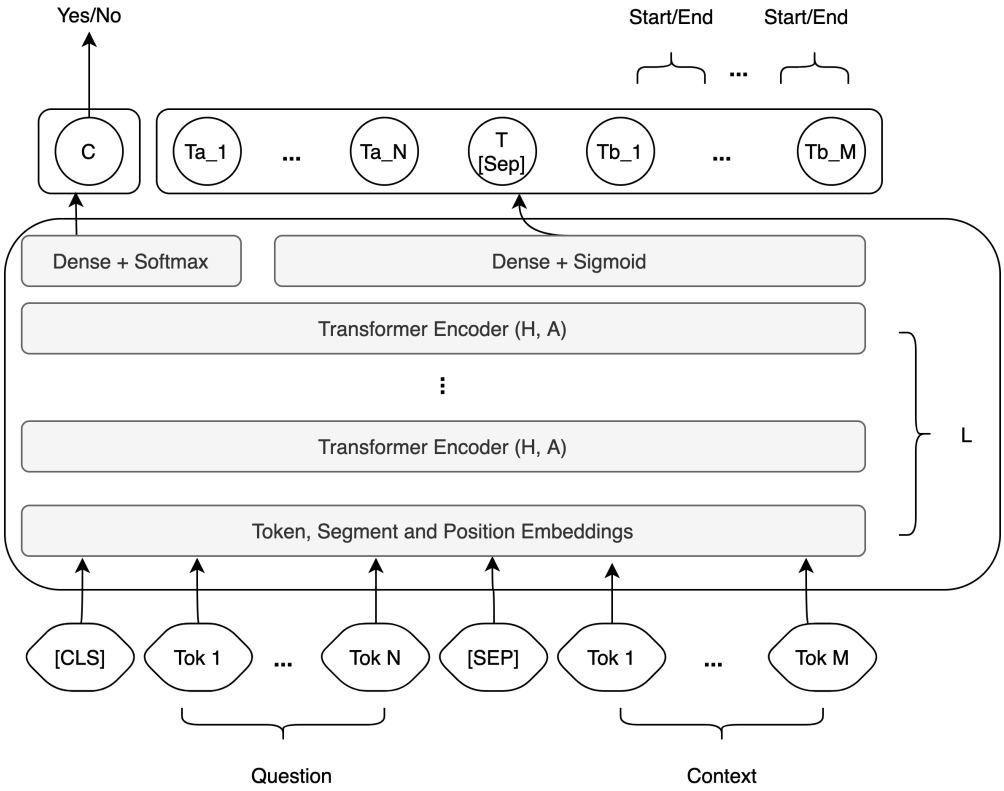

Figure 2. Extractor model architecture.

We define a training set data point as a four-tuple (d, s, e, yn), where d is a document containing the answers, s, e are indices to the start and end of the text answer, and yn is the yes–no–none answer.

The loss of our model is

$$L = \log p(s, e, yn|d)$$

$$L = \frac{1}{3}(\log ps(s|d) + \log pe(e|d) + \log pyn(yn|d))$$

where each probability p is defined as

$$ps(s|d) = \frac{1}{1 + \exp(-f_{start}(s, d; \theta))}$$

$$pe(e|d) = \frac{1}{1 + \exp(-f_{end}(e, d; \theta))}$$

$$pyn(yn|d) = \frac{\exp(-f_{yn}(yn, d; \theta))}{\sum yn' \exp(-f_{yn}(yn', d; \theta))'}$$

where θ is the base model parameters and f_{start}, f_{end}, f_{yn} represent three outputs from the last layer of the model.

At inference, we pass through all text from each document and return all start and end indices with scores higher than a threshold (implementation details are presented in Section 5.2). We used F1 and exact match (EM) metrics to evaluate our extractor models.

5. Experiments

We experimented with two information-retrieval systems for retrievers and seven models for extractors.

5.1. Retriever Experiments

In our experiments, we used pretrained and ready-to-use information-retrieval systems. Table 3 demonstrates the results of these experiments. We found that Amazon Kendra's semantic search is far superior to a simple keyword search; therefore, we chose Amazon Kendra as our retriever. Given a question, we used Amazon Kendra to find the documents that may contain the answer, then we passed the top nine documents to our extractor. Given that these information-retrieval systems are easy to use and easy to maintain and they are adequately performant, we chose to use these in our solution and focus our efforts on building a custom-built extractor. Section 6 states our plans for the next iteration of this solution which includes creating a custom-built retriever.

Table 3. Retrievers' performance.

Retriever	*P*@1	*P*@3	*P*@5	*P*@7	*P*@9	*P*@13	*P*@22	*P*@30	*P*@40	*P*@60
Whoosh	0.05	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06
Kendra	0.66	0.79	0.86	0.87	0.9	0.91	0.92	0.93	0.94	0.95

5.2. Extractor Experiments

Regarding the extractors, we found that the bigger the base model, the better the performance. We used popular pretrained BERT-based language models as described in Section 4.2 and fine-tuned these models on SQuAD1.1 and SQuAD2.0 [6] along with Natural Questions datasets [7]. We trained the models by minimizing loss L from Section 4.2.2 with the AdamW optimizer [18] with a batch size of eight. We used 0.5 as the threshold for start and end indices. Then, we tested our models against the AWS documentation dataset (Section 3.1) while using Amazon Kendra as the retriever. Our final results are shown in Table 4.

Table 4. End-to-end performance with Amazon Kendra as the retriever. L and H are hyperparameters for the extractor's base model.

Extractor Base Model	L	H	F1	EM
BERT Tiny	2	128	0.128	0.09
RoBERTa Base	12	768	0.154	0.09
DistilBERT	12	768	0.158	0.08
AlBERT Base	12	768	0.199	0.11
BERT Base	12	768	0.245	0.16
BERT Large	24	1024	0.247	0.16
AlBERT XXL	12	4096	0.422	0.39

5.3. Error Analysis

To better understand our solution and perform an error analysis, we manually analyzed our solution's predicted answers to the AWS Documentation dataset. We identified four categories of error in the results: retriever error, partial answer, table extraction error, and wrong prediction (see Table 5).

5.3.1. Exact Matches

The model provides an exact match for 39% of questions. Generally, these questions have a clear answer in the source document coming from a continuous block of text. Our solution works best in this category primarily because our training dataset mostly consists of factual questions and answers that can be extracted from a continuous block of text. For example, for the question "Can I run my AWS Lambda in a VPC?", our solution correctly predicted the answer as "You can configure a Lambda function to connect to private subnets in a virtual private cloud (VPC) in your AWS account". This answer is clearly stated in the document and the predicted answer had one start and one end token index. Another example in this category of questions is "In Amazon SageMaker, what is the internetwork communication protocol?", for which the solution correctly predicted "TLS 1.2".

5.3.2. Retriever Errors

As discussed in Section 5.1, we used Amazon Kendra as our retriever to obtain the top nine documents to pass to our extractor. Our retriever has an error rate of 10% which affects our end-to-end performance. Undeniably, if we cannot find the right document that contains the answer, we cannot find the correct answer. Improving our retriever's performance will enhance the overall performance of our solution. We are planning to perform experiments on this subject in our future works.

5.3.3. Partial Answers

For 7% of questions, our solution was not able to find an exact match; however, it was able to find parts of the answer. These questions have answers that should be extracted from different locations of a document, and our solution was not able to find all correct answer spans. For example, for the question "What are the Amazon RDS storage types?", the correct answer is "General Purpose SSD, Provisioned IOPS, and Magnetic", but our solution predicted the answer as "Provisioned IOPS", which is partly correct, but not entirely. These types of questions and answers were underrepresented in our training dataset. We are planning to train our models with more data in this category in our future work.

5.3.4. Table Extraction Errors

For 24% of questions, our solution was not able to extract the right information from a table in the source document. For example, for the question "What is the instance store type for d2.xlarge?", our solution was not able to extract the correct answer, which is "HDD", from a table within the document. In our training dataset, all of the questions and answers were presented in plain text and there were no examples with tables. In our future work, we will explore more datasets along with different approaches to tackle this challenge.

5.3.5. Wrong Predictions

For 20% of questions, our solution was not able to find any part of the answer. These questions and answers were challenging in nature, even for a human. For example, for the question "Is Administrator an Amazon Forecast reserved field name?", the correct answer is "No, Administrator is not an Amazon Forecast reserved field name", but our solution was not able to predict the correct answer. To answer this question correctly, a solution must read the document entirely and check for the "Administrator" word in the reserved field name. Because the word does not exist in the list, the answer should be "no". We hypothesize that if we use a generative extractor, we might be able to better answer these types of questions. We are planning to investigate this type of questions and answers in our future works.

Table 5. Manual error analysis results on AWS Documentation dataset.

Category	Percentage
Retriever error	10%
Partial answer	7%
Table extraction	24%
Wrong prediction	20%
Exact match	39%

6. Limitations and Future Work

Our solution has a number of limitations. Below, we describe some of these and suggest directions for future work. We were able to achieve 42% F1 and 39% EM for our test dataset due to the challenging nature of zero-shot open-book problems. The performance of the solution proposed in this article is fair if tested against technical software documentation. However, it needs to be improved before finding use in real-world software products. Additionally, more testing and evaluation is needed if we want to thoroughly assess the applicability of this solution in other domains (e.g., medical, financial, or laws and regulations). We are planning to experiment and evaluate this solution in multiple domains and compare it to fine-tuned (not zero-shot) models in our future work.

Furthermore, the solution performs better if the answer can be extracted from a continuous block of text from the document. The performance drops if the answer is extracted from several different locations in a document. Moreover, all questions had a clear answer in the AWS documentation dataset, which is not always the case in the real world. As our proposed solution always returns an answer to any question, it fails to recognize if a question cannot be answered.

For future work, we plan to experiment with custom-built retrievers similar to DPR [34], as well as generative models, such as GPT-2 [35] and GPT-3 [5], with a wider variety of text in pretraining to improve the F1 and EM scores presented in this article.

7. Conclusions

In this paper, we presented a new solution for zero-shot open-book QA with a retriever–extractor architecture to answer natural language questions from an available set of documents. We showed that ready-to-use information-retrieval systems can be used to find the right document and pass it to extractors. We demonstrated a custom-built transformer model to extract the answer within a set of documents. We experimented with several BERT-based models to identify the best-performing model for the task. With this solution, we were able to achieve 42% F1 and 39% EM with no domain-specific labeled data. We hope this new dataset and solution helps researchers create better solutions for zero-shot open-book use cases in similar real-world environments.

Author Contributions: S.G.: Conceptualization, methodology, software, investigation, writing—original draft. M.N.: Writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: The contributions of Sia Gholami and Mehdi Noori were funded by Amazon Web Services.

Institutional Review Board Statement: Not Applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data and code supporting the conclusions of this article are available at https://github.com/siagholami/zero-shot-open-book-qa, accessed on 18 December 2021.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Voorhees, E.M. The TREC-8 question answering track report. *Trec. Citeseer* **1999**, *99*, 77–82.
2. Moldovan, D.; Harabagiu, S.; Pasca, M.; Mihalcea, R.; Girju, R.; Goodrum, R.; Rus, V. The structure and performance of an open-domain question answering system. In Proceedings of the 38th annual meeting of the Association for Computational Linguistics, Hong Kong, China, 10–12 October 2000; pp. 563–570.
3. Brill, E.; Dumais, S.; Banko, M. An analysis of the AskMSR question-answering system. In Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002), Philadelphia, PA, USA, 6–7 July 2002; pp. 257–264.
4. Ferrucci, D.; Brown, E.; Chu-Carroll, J.; Fan, J.; Gondek, D.; Kalyanpur, A.A.; Lally, A.; Murdock, J.W.; Nyberg, E.; Prager, J.; et al. Building Watson: An overview of the DeepQA project. *AI Mag.* **2010**, *31*, 59–79. [CrossRef]
5. Brown, T.B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language models are few-shot learners. *arXiv* **2020**, arXiv:2005.14165.
6. Rajpurkar, P.; Zhang, J.; Lopyrev, K.; Liang, P. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Austin, TX, USA, 1–5 November 2016; pp. 2383–2392.
7. Kwiatkowski, T.; Palomaki, J.; Redfield, O.; Collins, M.; Parikh, A.; Alberti, C.; Epstein, D.; Polosukhin, I.; Devlin, J.; Lee, K.; et al. Natural questions: a benchmark for question answering research. *Trans. Assoc. Comput. Linguist.* **2019**, *7*, 453–466. [CrossRef]
8. Joshi, M.; Choi, E.; Weld, D.S.; Zettlemoyer, L. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Vancouver, BC, Canada, 12–15 July 2017; Volume 1: Long Papers, pp. 1601–1611.
9. Khashabi, D.; Chaturvedi, S.; Roth, M.; Upadhyay, S.; Roth, D. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Melbourne, Australia, 15–20 July 2018; Volume 1 (Long Papers), pp. 252–262.
10. Richardson, M.; Burges, C.J.; Renshaw, E. Mctest: A challenge dataset for the open-domain machine comprehension of text. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, WA, USA, 18–21 October 2013; pp. 193–203.
11. Lai, G.; Xie, Q.; Liu, H.; Yang, Y.; Hovy, E. RACE: Large-scale ReAding Comprehension Dataset From Examinations. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, 9–11 September 2017; pp. 785–794.
12. Reddy, S.; Chen, D.; Manning, C.D. Coqa: A conversational question answering challenge. *Trans. Assoc. Comput. Linguist.* **2019**, *7*, 249–266. [CrossRef]
13. Choi, E.; He, H.; Iyyer, M.; Yatskar, M.; Yih, W.T.; Choi, Y.; Liang, P.; Zettlemoyer, L. QuAC: Question Answering in Context. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 30 October–2 November 2018; pp. 2174–2184.
14. Tafjord, O.; Clark, P.; Gardner, M.; Yih, W.T.; Sabharwal, A. Quarel: A dataset and models for answering questions about qualitative relationships. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 7063–7071.
15. Mitra, A.; Clark, P.; Tafjord, O.; Baral, C. Declarative question answering over knowledge bases containing natural language text with answer set programming. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 3003–3010.
16. Seo, M.; Kembhavi, A.; Farhadi, A.; Hajishirzi, H. Bidirectional attention flow for machine comprehension. *arXiv* **2016**, arXiv:1611.01603.
17. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*; NIPS: Long Beach, CA, USA, 4–9 December 2017; pp. 5998–6008.
18. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. 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, Minneapolis, MN, USA, 2–7 June 2019; Volume 1 (Long and Short Papers), pp. 4171–4186.
19. Zhang, S.; Zhang, X.; Wang, H.; Cheng, J.; Li, P.; Ding, Z. Chinese Medical Question Answer Matching Using End-to-End Character-Level Multi-Scale CNNs. *Appl. Sci.* **2017**, *7*, 767. [CrossRef]
20. Boban, I.; Doko, A.; Gotovac, S. Improving Sentence Retrieval Using Sequence Similarity. *Appl. Sci.* **2020**, *10*, 4316. [CrossRef]

21. Pota, M.; Esposito, M.; De Pietro, G.; Fujita, H. Best Practices of Convolutional Neural Networks for Question Classification. *Appl. Sci.* **2020**, *10*, 4710. [CrossRef]
22. Sarhan, I.; Spruit, M. Can We Survive without Labelled Data in NLP? Transfer Learning for Open Information Extraction. *Appl. Sci.* **2020**, *10*, 5758. [CrossRef]
23. Mutabazi, E.; Ni, J.; Tang, G.; Cao, W. A Review on Medical Textual Question Answering Systems Based on Deep Learning Approaches. *Appl. Sci.* **2021**, *11*, 5456. [CrossRef]
24. Jin, D.; Pan, E.; Oufattole, N.; Weng, W.H.; Fang, H.; Szolovits, P. What Disease Does This Patient Have? A Large-Scale Open Domain Question Answering Dataset from Medical Exams. *Appl. Sci.* **2021**, *11*, 6421. [CrossRef]
25. Phakmongkol, P.; Vateekul, P. Enhance Text-to-Text Transfer Transformer with Generated Questions for Thai Question Answering. *Appl. Sci.* **2021**, *11*, 267. [CrossRef]
26. Ali, W.; Zuo, W.; Ali, R.; Zuo, X.; Rahman, G. Causality Mining in Natural Languages Using Machine and Deep Learning Techniques: A Survey. *Appl. Sci.* **2021**, *11*, 64. [CrossRef]
27. Banerjee, P.; Pal, K.K.; Mitra, A.; Baral, C. Careful Selection of Knowledge to Solve Open Book Question Answering. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; pp. 6120–6129.
28. Yasunaga, M.; Ren, H.; Bosselut, A.; Liang, P.; Leskovec, J. QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Online, 6–11 June 2021; pp. 535–546.
29. Hobbs, J.R. Abduction in natural language understanding. In *Handbook of Pragmatics*; Wiley: Hoboken, NJ, USA, 2004; pp. 724–741.
30. Pérez-Agüera, J.R.; Arroyo, J.; Greenberg, J.; Iglesias, J.P.; Fresno, V. Using BM25F for semantic search. In Proceedings of the 3rd International Semantic Search Workshop, Raleigh, NC, USA, 26–30 April 2010; pp. 1–8.
31. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv* **2019**, arXiv:1907.11692.
32. Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; Soricut, R. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In Proceedings of the International Conference on Learning Representations, Addis Ababa, Ethiopia, 26–30 April 2020.
33. Sanh, V.; Debut, L.; Chaumond, J.; Wolf, T. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *arXiv* **2019**, arXiv:1910.01108.
34. Karpukhin, V.; Oguz, B.; Min, S.; Lewis, P.; Wu, L.; Edunov, S.; Chen, D.; Yih, W.T. Dense Passage Retrieval for Open-Domain Question Answering. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Virtual Conference, 16–20 November 2020; pp. 6769–6781.
35. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language models are unsupervised multitask learners. *OpenAI Blog* **2019**, *1*, 9.