# End-to-End Open-Domain Question Answering with BERTserini

**Wei Yang,**[*1,2] **Yuqing Xie,**[*1,2] **Aileen Lin,**[2] **Xingyu Li,**[2]
**Luchen Tan,**[2] **Kun Xiong,**[2] **Ming Li,**[1,2] **Jimmy Lin**[1,2]
[1] David R. Cheriton School of Computer Science, University of Waterloo
[2] RSVP.ai



## Abstract

We demonstrate an end-to-end question answering system that integrates BERT with the open-source Anserini information retrieval toolkit. In contrast to most question answering and reading comprehension models today, which operate over small amounts of input text, our system integrates best practices from IR with a BERT-based reader to identify answers from a large corpus of Wikipedia articles in an end-to-end fashion. We report large improvements over previous results on a standard benchmark test collection, showing that fine-tuning pretrained BERT with SQuAD is sufficient to achieve high accuracy in identifying answer spans.

## 1 Introduction

BERT (Devlin et al., 2018), the latest refinement of a series of neural models that make heavy use of pretraining (Peters et al., 2018; Radford et al., 2018), has led to impressive gains in many natural language processing tasks, ranging from sentence classification to question answering to sequence labeling. In this demonstration, we integrate BERT with the open-source Anserini IR toolkit to create BERTserini, an end-to-end open-domain question answering (QA) system.

Unlike most QA or reading comprehension models, which are best described as rerankers or extractors since they assume as input relatively small amounts of text (an article, top $k$ sentences or passages, etc.), our system operates directly on a large corpus of Wikipedia articles. We integrate best practices from the information retrieval community with BERT to produce an end-to-end system, and experiments on a standard benchmark test collection show large improvements over previous work. Our results show that fine-tuning pretrained BERT with SQuAD (Rajpurkar et al., 2016) is sufficient to achieve high accuracy in identifying answer spans. The simplicity of this design is one major feature of our architecture. We have deployed BERTserini as a chatbot that users can interact with on diverse platforms, including laptops and mobile phones.

## 2 Background and Related Work

While the origins of question answering date back to the 1960s, the modern formulation can be traced to the Text Retrieval Conferences (TRECs) in the late 1990s (Voorhees and Tice, 1999). With roots in information retrieval, it was generally envisioned that a QA system would comprise pipeline stages that selected increasingly finer-grained segments of text (Tellex et al., 2003): document retrieval to identify relevant documents from a large corpus, followed by passage ranking to identify text segments that contain answers, and finally answer extraction to identify the answer spans.

As NLP researchers became increasingly interested in QA, they placed greater emphasis on the later stages of the pipeline to emphasize various aspects of linguistic analysis. Information retrieval techniques receded into the background and became altogether ignored. Most popular QA benchmark datasets today—for example, TrecQA (Yao et al., 2013), WikiQA (Yang et al., 2015), and MSMARCO (Bajaj et al., 2016)—are best characterized as answer selection tasks. That is, the system is given the question as well as a candidate list of sentences to choose from. Similarly, reading comprehension datasets such as SQuAD (Rajpurkar et al., 2016) eschew retrieval entirely, since there is only a single document from which to extract answers.

In contrast, what we refer to as "end-to-end" question answering begins with a large corpus of

[*] equal contribution

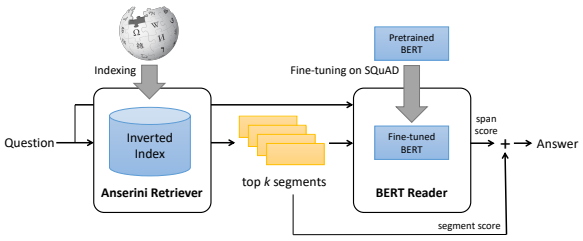


Figure 1: Architecture of BERTserini.

documents. Since it is impractical to apply inference exhaustively to all documents in a corpus with current models (mostly based on neural networks), this formulation necessarily requires some type of term-based retrieval technique to restrict the input text under consideration—and hence an architecture quite like systems from over a decade ago. Recently, there has been a resurgence of interest in this task, the most notable of which is Dr.QA (Chen et al., 2017). Other recent papers have examined the role of retrieval in this end-to-end formulation (Wang et al., 2017; Kratzwald and Feuerriegel, 2018; Lee et al., 2018), some of which have, in essence, rediscovered ideas from the late 1990s and early 2000s.

For a wide range of applications, researchers have recently demonstrated the effectiveness of neural models that have been pretrained on a language modeling task (Peters et al., 2018; Radford et al., 2018); BERT (Devlin et al., 2018) is the latest refinement of this idea. Our work tackles end-to-end question answering by combining BERT with Anserini, an IR toolkit built on top of the popular open-source Lucene search engine. Anserini (Yang et al., 2017, 2018) represents recent efforts by researchers to bring academic IR into better alignment with the practice of building real-world search applications, where Lucene has become the *de facto* platform used in industry. Through an emphasis on software engineering and regression testing for replicability, Anserini codifies IR best practices today. Recently, Lin (2018) showed that a well-tuned Anserini implementation of a query expansion model proposed over a decade ago still beats two recent neural models for document ranking. Thus, BERT and Anserini represent solid foundations on which to build an end-to-end question answering system.

## 3 System Architecture

The architecture of BERTserini is shown in Figure 1 and is comprised of two main modules, the Anserini retriever and the BERT reader. The retriever is responsible for selecting segments of text that contain the answer, which is then passed to the reader to identify an answer span. To facilitate comparisons to previous work, we use the same Wikipedia corpus described in Chen et al. (2017) (from Dec. 2016) comprising 5.08M articles. In what follows, we describe each module in turn.

### 3.1 Anserini Retriever

For simplicity, we adopted a single-stage retriever that directly identifies segments of text from Wikipedia to pass to the BERT reader—as opposed to a multi-stage retriever that first retrieves documents and then ranks passages within. However, to increase flexibility, we experimented with different granularities of text at indexing time:

**Article**: The 5.08M Wikipedia articles are directly indexed; that is, an article is the unit of retrieval.

**Paragraph**: The corpus is pre-segmented into 29.5M paragraphs and indexed, where each paragraph is treated as a "document" (i.e., the unit of retrieval).

**Sentence**: The corpus is pre-segmented into 79.5M sentences and indexed, where each sentence is treated as a "document".

At inference time, we retrieve $k$ text segments (one of the above conditions) using the question as a "bag of words" query. We use a post-v0.3.0 branch of Anserini,[1] with BM25 as the ranking function (Anserini's default parameters).

### 3.2 BERT Reader

Text segments from the retriever are passed to the BERT reader. We use the model in Devlin et al. (2018), but with one important difference: to allow comparison and aggregation of results from different segments, we remove the final softmax layer over different answer spans.

Our BERT reader is based on Google's reference implementation[2] (TensorFlow 1.12.0). For training, we begin with the BERT-Base model (uncased, 12-layer, 768-hidden, 12-heads, 110M parameters) and then fine tune the model on the training set of SQuAD (v1.1). All inputs to the reader are padded to 384 tokens; the learning rate is set to $3 \times 10^{-5}$ and all other defaults settings are used.

At inference time, for retrieved articles, we apply the BERT reader paragraph by paragraph. For

[1] http://anserini.io/

[2] https://github.com/google-research/bert

| Model | EM | R | F1 |
|---|---|---|---|
| Dr.QA (Chen et al., 2017) | 27.1 | 77.8 | - |
| Dr.QA + Fine-tune | 28.4 | - | - |
| Dr.QA + Multitask | 29.8 | - | - |
| $R^3$ (Wang et al., 2017) | 29.1 | - | 37.5 |
| Kratzwald and Feuerriegel (2018) | 29.8 | - | - |
| Par. R. (Lee et al., 2018) | 28.5 | 83.1 | - |
| Par. R. + Answer Agg. | 28.9 | - | - |
| Par. R. + Full Agg. | 30.2 | - | - |
| BERTserini (Article, $k = 5$) | 19.1 | 63.1 | 25.9 |
| BERTserini (Paragraph, $k = 29$) | 36.6 | 75.0 | 44.0 |
| BERTserini (Sentence, $k = 78$) | 34.0 | 67.5 | 41.0 |
| BERTserini (Paragraph, $k = 100$) | **38.6** | **85.8** | **46.1** |

Table 1: Results on SQuAD development questions.

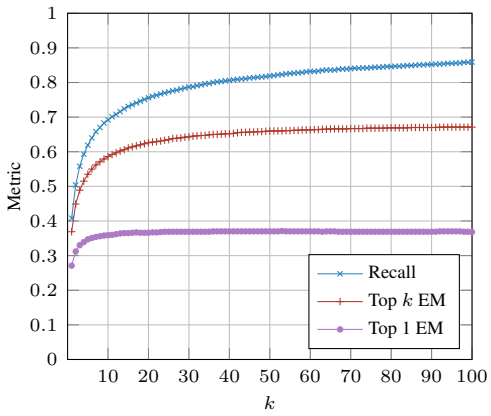


Figure 2: Model effectiveness with different numbers of retrieved paragraphs.

retrieved paragraphs, we apply inference over the entire paragraph. For retrieved sentences, we apply inference over the entire sentence. In all cases, the reader selects the best text span and provides a score. We then combine the reader score with the retriever score via a simple linear interpolation:

$$S = (1 - \mu) \cdot S_{\text{Anserini}} + \mu \cdot S_{\text{BERT}}$$

where $\mu \in [0, 1]$ is a hyperparameter. We tune $\mu$ on 1000 randomly-selected question-answer pairs from the SQuAD training set, considering all values in tenth increments.

## 4 Experimental Results

We adopt exactly the same evaluation methodology as Chen et al. (2017), which was also used in subsequent work. Test questions come from the development set of SQuAD; since our answers come from different texts, we only evaluate with respect to the SQuAD answer spans (i.e., the passage context is ignored). Our evaluation metrics are also the same as Chen et al. (2017): exact match (EM) score, recall (R), and F1 score.

Our main results are shown in Table 1, where we report metrics with different Anserini retrieval conditions (article, paragraphs, and sentences). We compare article retrieval at $k = 5$, paragraph retrieval at $k = 29$, and sentence retrieval at $k = 78$. The article setting matches the retrieval condition in Chen et al. (2017). The values of $k$ for the paragraph and sentence conditions are selected so that the reader considers approximately the same amount of text: each paragraph contains 2.7 sentences on average, and each article contains 5.8 paragraphs on average. The table also copies results from previous work for comparison.

We see that article retrieval underperforms paragraph retrieval by a large margin: the reason, we believe, is that articles are long and contain many non-relevant sentences that serve as distractors to the BERT reader. Sentences perform reasonably but not as well as paragraphs because they often lack the context for the reader to identify the answer span. Paragraphs seem to represent a "sweet spot", resulting in a large improvement in exact match score over previously-published results.

Our next experiment examined the effects of varying $k$, the number of text segments considered by the BERT reader. Here, we focus only on the paragraph condition, with $\mu = 0.5$ (the value learned via cross validation). Figure 2 plots three metrics with respect to $k$: recall, top $k$ exact match, and top exact match. Recall measures the fraction of questions for which the correct answer appears in *any* retrieved segment. Top $k$ exact match represents a lenient condition where the system receives credit for a correctly-identified span in *any* retrieved segment. Finally, top exact match is evaluated with respect to the top scoring span, comparable to the results reported in Table 1. Scores for the paragraph condition at $k = 100$ are also reported in the table: we note that the exact match score is eight points higher than the previously-published best result that we are aware of.

We see that, as expected, scores increase with larger $k$ values. However, the top exact match score doesn't appear to increase much after around $k = 10$. The top $k$ exact match score continues growing a bit longer but also reaches saturation. Recall appears to continue growing all the way up to $k = 100$, albeit more slowly as $k$ increases. However, the BERT reader is unable to take ad-

vantage of these additional answer segments.

These curves also provide a failure analysis: The top recall curve (in blue) represents the upper bound with the current Anserini retriever. The gap between that and the top $k$ exact match curve (in red) quantifies the room for improvement with the BERT reader. The gap between the red curve and the bottom top exact match curve (in purple) represents cases where BERT *did* identify the correct answer, but not as the top scoring span. Based on the plot, this seems to be the biggest area for improvement—this gap can be characterized as failures in accurate scoring or score aggregation, suggesting that our current approach (weighted interpolation between the BERT and Anserini scores) is insufficient. We are exploring reranking models that are capable of integrating more relevance signals. This analysis also shows that the Anserini retriever is performing quite well: At $k = 100$, it is able to return at least one relevant paragraph around 86% of the time. Recall can be further boosted, we believe, via query expansion techniques. There is also quite a bit of room for improvement with the BERT reader, highlighting potential advances that can come from better modeling.

## 5 Demonstration

We have deployed BERTserini as a chatbot that users can interact with on multiple platforms. The entire backend is written in Python, which connects to the JVM (since Anserini is written in Java) via the Pyjnius library. We expose a REST API via Flask, and the current web-based interface is written in the React JavaScript library. A screenshot is shown in Figure 3. The current interface uses the paragraph indexing condition, but we return only the sentence containing the answer identified by the BERT reader. The answer span is highlighted in the response. In the screenshot we can see the diversity of questions that BERTserini can handle—different types of named entities as well as queries whose answers are not noun phrases.

One important consideration in an operational system is the latency of the responses. Informed by the analysis in Figure 2, in our demonstration system we set $k = 10$ under the paragraph condition. While this does not give us the maximum possible accuracy, it represents a good cost/quality tradeoff. To quantify processing time, we randomly selected 100 questions from SQuAD and


Figure 3: A screenshot of BERTserini in our current chatbot interface. The first question is from SQuAD; the remaining three questions illustrate the range of questions that the system can answer.

recorded average latencies; measurements were taken on a machine with an Intel Xeon E5-2620 v4 CPU (2.10GHz) and a Tesla P40 GPU. Anserini retrieval (on the CPU) averages 0.5s per question, while BERT processing time (on the GPU) averages 0.18s per question.

## 6 Conclusion

We introduce BERTserini, our end-to-end open domain question answering system that integrates BERT and the Anserini IR toolkit. With a simple two-stage pipeline architecture, we are able to achieve large improvements over previous systems. Error analysis points to room for improvement in retrieval, answer extraction, and answer aggregation—all of which represent ongoing efforts. In addition, we are also interested in expanding the multilingual capabilities of our system.

## References


Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2016. MS MARCO: A human generated MAchine Reading COmprehension dataset. *arXiv:1611.09268*.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*.

Bernhard Kratzwald and Stefan Feuerriegel. 2018. Adaptive document retrieval for deep question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 576–581.

Jinhyuk Lee, Seongjun Yun, Hyunjae Kim, Miyoung Ko, and Jaewoo Kang. 2018. Ranking paragraphs for improving answer recall in open-domain question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 565–569.

Jimmy Lin. 2018. The neural hype and comparisons against weak baselines. *SIGIR Forum*, 52(2):40–51.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. Technical report.

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

Stefanie Tellex, Boris Katz, Jimmy Lin, Gregory Marton, and Aaron Fernandes. 2003. Quantitative evaluation of passage retrieval algorithms for question answering. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2003)*, pages 41–47, Toronto, Ontario, Canada.

Ellen M. Voorhees and Dawn M. Tice. 1999. The TREC-8 Question Answering Track evaluation. In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, pages 83–106, Gaithersburg, Maryland.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. 2017. $R^3$: Reinforced reader-ranker for open-domain question answering. *arXiv:1709.00023*.

Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the use of Lucene for information retrieval research. In *Proceedings of the 40th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2017)*, pages 1253–1256, Tokyo, Japan.

Peilin Yang, Hui Fang, and Jimmy Lin. 2018. Anserini: Reproducible ranking baselines using Lucene. *Journal of Data and Information Quality*, 10(4):Article 16.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018.

Xuchen Yao, Benjamin Van Durme, Chris Callison-burch, and Peter Clark. 2013. Answer extraction as sequence tagging with tree edit distance. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 858–867.