PtEQA: Domain Specific Question Answering with Document Extraction Utilizing Pre-trained Weights

Mohan Zhang, Sudeep Singh, Pawel Maciszewski
ExxonMobil, University of Toronto
505 Quarry Park Blvd Calgary, 27 King's College Circle Toronto
zhangmo4@cs.toronto.edu, sudeep.singh@exxonmobil.com, pawel.maciszewski@exxonmobil.com

Abstract

This paper proposes a domain-specific Question and Answering model: given a text corpus, this model can answer questions with regard to that corpus. The corpus may contain up to 10^7 documents, and those documents are not necessarily labeled or even structured. This model also provides a powerful chatbot-ready solution utilizing pre-trained weights: It reaches state-of-the-art performance before any training on the TriviaQA dataset. The model has two components: (1) Document Retriever, to retrieve the most relevant ones from all documents using bi-gram hashing and TF-IDF reweighting. (2) Document Reader, to extract the answer from previously retrieved documents using fine-tuned Bidirectional Encoder Representations from Transformers with pre-trained weights.

Keywords: chatbot, QA, Pretrained

1. Introduction

The demand for a Question Answering (QA) system for an arbitrary set of unlabeled and unstructured documents is growing fast. In industry, most of the time when we want to build a QA system or chatbot, we cannot provide the system with enough labeled data to be trained on, simply because data labeling requires time and labor. Traditionally, people use structured Knowledge Bases (KBs) like Freebase (Bollacker et al., 2008), DBpedia (Auer et al., 2007), or other novel inference models for knowledge representation and reasoning (Talmor and Berant, 2018), nevertheless, requires effort and resources to build up, as well as suffering from the inherited limitations from mapping pure text to logical forms, addressed by (Chen et al., 2014) and (Riedel et al., 2013). There are also complicated pipeline systems like (Ferrucci et al., 2010) to integrate multiple hand-crafted modules together for open domain question and answering systems, but they are also not suitable for QA on unstructured documents since these type of solutions are limited by human design. But the good news is, there are quite a few labeled QA datasets proposed in recent years. For example, CoQA (Reddy et al., 2019); a conversational QA dataset contains 127,000+ questions with answers collected from 8000+ conversations. MS MACRO (Nguyen et al., 2016) is a human-generated machine reading comprehension dataset, for which the answers are human generated. The context passages are extracted from real documents using the latest Bing search engine. MultiRC (Khashabi et al., 2018) is a dataset of short paragraphs and multi-sentence questions that can be answered from the content of the paragraph. The NewsQA dataset (Trischler et al., 2017) is a reading comprehension dataset of over 100,000 human-generated question-answer pairs from over 10,000 news articles from CNN, with answers consisting of spans of text from the corresponding articles.

In this paper, we train our model on the Stanford Question Answering Dataset (SQuAD), which is a reading comprehension dataset, consisting of questions posed by crowdworkers on a set of Wikipedia articles. The answer to every question is a segment of text (span) from the corresponding reading passage. Our work is currently Based on SQuAD 1.1 (Rajpurkar et al., 2016). Our model proposes a way to utilize the SQuAD labeled data to train our Document Reader for its capabilities of Reading Comprehension and Language Understanding, thus we can directly use those pre-trained weights for a large chunk of unlabeled documents or further fine-tune on a small set of labeled documents.

2. Related Work

Our work is inspired by the Open Domain Question Answering system DrQA proposed by (Chen et al., 2017). In their paper, they proposed a way to tackle down the "machine reading at scale" problem, especially for Question and Answering on a large unlabeled corpus, by first filtering the corpus with an efficient Document Retriever and then extracting the answers from filtered documents with a pre-trained Document Reader.

2.1. Document Retrieval

Document Retrieval with respect to a free form user input itself has a long history, though the query is not necessarily a question. As described in the book (Krishna, 1992), relational database queries and knowledge base inferences are always options if we are seeking for efficiency, but at the sacrifice of building up the customized database or knowledge base system. We can also create annotations and create a system for parsing as mentioned in (Gaizauskas and Wilks, 1998), but it again needs to preprocess the whole corpus. In order to avoid the trouble of any pre-processing and efficiently deal with any form of unstructured plain text documents, we choose to use the search-engine-like document retrieving method, similar to the one used in (Chen et al., 2017). More details are explained in section 3.1. Document Retriever.

2.2. Answer Extraction

Here, we will only discuss neural reading comprehension for Answer Extraction on a short document, since we have already narrowed down the problem to a small subset of

documents with Document Retriever.
Traditionally, there are some knowledge representation and reasoning ways to build up knowledge bases then do machine reading and comprehension with textual entailment (Gleize, 2016a), but those techniques are either not accurate or they require a ton of labeling effort. Now the generous contribution to open-sourced short-paragraph QA dataset like SQuAD (Rajpurkar et al., 2016), MS MACRO (Nguyen et al., 2016) and NewsQA (Trischler et al., 2017), enables a rapid development of deep neural networks, from CNN (Gleize, 2016b), RNN (Joshi et al., 2017), LSTM (Cheng et al., 2016) to attention-based-Transformer (GPT-2). In this paper, we use a fine-tuned BERT (Devlin et al., 2018) to retrieve answers. More details are explained in section 3.2. Document Reader.

2.3. Why Not Using GPT-2?

As compared to BERT (Devlin et al., 2018), GPT-2 (Radford et al., 2019) is claimed to be a more powerful pre-trained model. However, we simply don't have enough processing power to pre-train it for now.

3. Model Architecture

3.1. Document Retriever

Given a question, for a large scale corpus that contains 10^a documents, the usual constraint is $3 \leq a \leq 7$(this method also works fairly good for a = 1). The first step is to locate relative documents in an efficient manner and the following step is to extract answers from selected documents using an accurate method. Generally machine reading at a large scale with deep neural networks, in order to extract answers, is time-consuming.
Hence this proposed solution uses a non-machine-learning document retriever, in order to find a small subset (less than 10 articles) of articles conditioned on the given question.

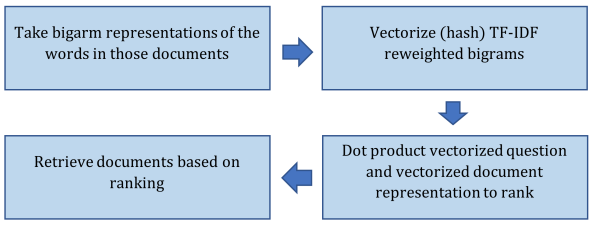

Figure 1: Architecture of Document Retriever

As per Figure 1, the document retriever first takes bigarms out of all the documents in the corpus, then those bigarms are hashed to 2^{24} bins using murmur3 hashing (Weinberger et al., 2009). In this way, each of the document has its own vector representation by its bigarms, which are reweighted by TF-IDF. To retrieve documents for a given question, we need to rank the relativity of those documents with respect to that question. This time we take bigarms of the given question, hash it and get a vector representation. Then for each of the documents, we dot product its vector representation and the given question's vector representation. The result obtained is regarded as the rank.
The number of articles selected this way is decided dynamically. Based on the EMNLP 2018 paper (Kratzwald and Feuerriegel, 2018), the number of documents retrieved by the Document Retriever depends on the number of documents in the whole corpus, ranges from 1 to 10, in order to achieve the best sensitivity with acceptable precision. More details are explained in section 3.3. Retrieve Number and Document Tailor. Also, we assume here that the input corpus and the training corpus (The SQuAD dataset here in our paper) shares similar document-length distribution.

3.2. Document Reader

Our document reader is based on the paper *"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"* (Devlin et al., 2018). The document reader is a 12-layer and 12-attention-head BERT with 768 hidden units (the same as $BERT_{BASE}$'s size). We took the case-insensitive weights for English from Google, which was trained on Wikipedia and BookCorpus (Zhu et al., 2015) with a type of cloze task (Masked LM) and the next sentence prediction task. We then fine-tune it on SQuAD 1.1 (Rajpurkar et al., 2016) for the model to gain question and answering capability. The 110-million-parameter weights we gained after these two steps were then being used to test the model on TriviaQA in section 4. Experiments. We also tested the proposed solution on internal documents in section 4.3. Internal Tests.

3.3. Document Tailoring and Retrieval

In this section, we are aiming to explain why we choose to set a maximum length and tailor a long document into several short documents in the input corpus.
Firstly, as suggested by the paper (Kratzwald and Feuerriegel, 2018), the Noise-Information trade-off in Document Retrieval should be calculated based on the length distribution of documents in the corpus. It is not realistic to retrain the Adaptive Document Retrieval model every time we have a new corpus (since the training process requires labeled question-answer pairs), and it is hard to control the medium and average length for the corpus. To tackle down the problem, we hereby set a hard maximum number of words limitation and truncate every document based on its sentence structure with sliding window approach options. For more details, please consult the code (SupportingScripts/DocumentsSegmentorandFillInFiles.py) of this model.
Secondly, if the document is too long and the Document Reader was trained on a corpus where only a few training documents have that length, then the Document Reader itself becomes biased and tends not to select an answer that appears in the later part of the document, which is also one of the reason why we choose to truncate long documents.

4. Experiments

As mentioned before, this model is designed specifically for domain-specific Question and Answering on a large and unlabeled set of documents. We choose to evaluate the model on the public dataset TriviaQA.
For this evaluation task, we did not use any of the training data provided by TriviaQA. Instead, we directly use the weights gained by pretraining as described in section 3.2. Document Reader. Furthermore, we use the entire 487,000

documents provided by TriviaQA as the corpus to retrieve answers from.

4.1. TriviaQA

As described in the paper *"TriviaQA: A Large Scale Dataset for Reading Comprehension"* (Joshi et al., 2017), TriviaQA is a reading comprehension dataset containing over 650K question-answer-evidence triples. TriviaQA includes 95K question-answer pairs authored by trivia enthusiasts and independently gathered evidence documents, six per question on average, that provides high-quality distant supervision for answering the questions.
There are over 487K evidence documents provided by TriviaQA. 74,070 of them are Wikipedia articles and 413,173 of them are web articles retrieved by Bing Web search API.

4.2. Results

The code is now available here: *anonymous repo*. All the results can be reproduced by executing the scripts inside this repository.
Below is our test result on the TriviaQA dataset. As mentioned in section 3.2. Document Reader, we use the full document set, 487K unlabelled documents from web search and Wikipedia, as the corpus (search base for Document Retriever) for all the evaluation. Then we inference on questions provided in verified-web-dev, verified-wikipeidia-dev and web-dev (those set of questions can be found either on the official website or can be downloaded here). Our model has a decent performance even **without any training** on the training set provided by TriviaQA, as shown below.

our test result:		
dataset name	exact match	f1
verified-wikipedia-dev	**68.553**	**78.611**
verified-web-dev	**68.550**	**76.705**
web-dev	63.023	71.151
state-of-the-art test result:	68.65	73.07

Table 1: The only things we picked from those datasets are questions and answers

Due to the design of our system, our evaluation procedure is slightly different from the evaluation.py provided by the TriviaQA official repository: Instead of picking one final answer for each question asked in the question set, we pick 7 potential answers among all the 487K documents for that question, then calculate both exact match and f1 score out of the best: Let $f1(m, n)$ be the f1 score between two strings m and n. For a question q_i, let the set $A_i = \{a_{i_1}, a_{i_2}, ..., a_{i_7}\}$ be the set of 7 potential answers, let gt_i be the ground-truth answer. Then, the exact match score for question q_i is defined as:

$$exactMatch_i = \begin{cases} 1, & \text{if } \exists j \in \{1, 2, ..., 7\} \\ & \text{such that } a_{i_j} == gt_i \\ 0, & \text{otherwise} \end{cases}$$

The f1 score for question q_i is defined as: $f1_i = \max_{j \in \{1,2,...,7\}} \{f1(a_{i_j}, gt_i)\}$.
The evaluation score in our evaluation procedure is slightly different from the original evaluation score, because we are picking out of the best, our model is inferring on the **whole set of evidence documents** and the purpose of our design is not to solve the tasks mentioned in the TriviaQA paper (Joshi et al., 2017). However, the scores can still serve as an evidence of performance, though direct comparison between them is not fair.
For details please read the code in SupportingScripts/run_eval_triviaQA.py.

4.3. Internal Tests

We also labeled a small set of internal documents on some specific areas, fine-tuned the model respectively and tested it on another set of internal documents of the same area. The test result seems promising even with small labeled dataset and trivial data augmentation techniques. We are planning to put the model into an application.

5. Future Work

Limited by our current computing power, our current model only uses a $BERT_{BASE}$-size document reader for now. We are planning to enlarge the size of our Document Reader to improve the performance.
We are continuously validating the model on internal datasets and finding more applications for our model, as well as preparing labeled documents for the model to be fine-tuned on.

5.1. Application

The main purpose for us to build this kind of Question and Answering model is to design an efficient and accurate information retrieval application. The application would be integrated with chat-bots so as to improve the efficiency of both knowledge management departments and employees. To be more specific, there are tons of documents in our database and no one can read through them all, we are utilizing this domain-specific question and answering capability to **reactivate the knowledge accumulated** through many years of the operation of our company. To serve this purpose, the proposed solution is designed to be able to deal with a huge number of documents as well as to be training-data-free (with pre-trained weights). After finishing the development and training processes, we tested the model with an internal corpus, the result turned out to be promising. Therefore, we are continuously working on the proposed solution to make it into an application.

6. Conclusion

After several rounds of evaluation and test against public and private datasets, our developed solution shows a strong domain specific question and answering capability: It can efficiently retrieve a tiny (2-10 documents) subset of relevant documents from a large and unstructured corpus, then accurately locate the answer simply with pre-trained weights. As compared to traditional RNN, LSTM, encoder-decoders and other complex pipeline systems that designed specifically to solve QA and IR (information retrieval) problems, the language model learned by BERT from cloze and next-sentence-prediction task can serve more universal purposes through knowledge transfer, because BERT already have an

"understanding" of the language itself before it is fine-tuned on any QA dataset. The model also provides a ready-to-go solution for domain-specific QA chat-bot, since a corpus is the only thing that needs to be provided.

The disadvantages of this model are, it performs poorly on those questions that require cross-source references and it cannot process any series of conversational questions, due to the design limitation.

7. Bibliographical References

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Ives, Z., and et al. (2007). Dbpedia: A nucleus for a web of open data. In *IN 6TH INTERNATIONAL SEMANTIC WEB CONFERENCE, BUSAN, KOREA*, pages 11–15. Springer.

Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. (2008). Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1247–1250, New York, NY, USA. ACM.

Chen, L., Feng, Y., Huang, S., Qin, Y., and Zhao, D. (2014). Encoding relation requirements for relation extraction via joint inference. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 818–827.

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

Cheng, J., Dong, L., and Lapata, M. (2016). Long Short-Term Memory-Networks for Machine Reading. *arXiv e-prints*, page arXiv:1601.06733, Jan.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv e-prints*, October.

Ferrucci, D. A., Brown, E. W., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A., Lally, A., Murdock, J. W., Nyberg, E., Prager, J. M., Schlaefer, N., and Welty, C. A. (2010). Building watson: An overview of the deepqa project. *AI Magazine*, 31(3):59–79.

Gaizauskas, R. J. and Wilks, Y. (1998). Information extraction: Beyond document retrieval. *Journal of Documentation*, 54:70–105.

Gleize, M. (2016a). *Textual Inference for Machine Comprehension*. Theses, Université Paris-Saclay, January.

Gleize, M. (2016b). *Textual Inference for Machine Comprehension*. Theses, Université Paris-Saclay, January.

Joshi, M., Choi, E., Weld, D., and Zettlemoyer, L. (2017). TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada, July. Association for Computational Linguistics.

Khashabi, D., Chaturvedi, S., Roth, M., Upadhyay, S., and Roth, D. (2018). Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *NAACL*.

Kratzwald, B. and Feuerriegel, S. (2018). Adaptive document retrieval for deep question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 576–581, Brussels, Belgium, October-November. Association for Computational Linguistics.

Krishna, S. (1992). *Introduction to database and knowledge-base systems*. World Scientific.

Nguyen, T., Rosenberg, M., Song, X., Gao, J., Tiwary, S., Majumder, R., and Deng, L. (2016). Ms marco: A human generated machine reading comprehension dataset. November.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners.

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

Reddy, S., Chen, D., and Manning, C. D. (2019). CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266, March.

Riedel, S., Yao, L., McCallum, A., and Marlin, B. M. (2013). Relation extraction with matrix factorization and universal schemas. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 74–84.

Talmor, A. and Berant, J. (2018). The web as a knowledge-base for answering complex questions. 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 641–651, New Orleans, Louisiana, June. Association for Computational Linguistics.

Trischler, A., Wang, T., Yuan, X., Harris, J., Sordoni, A., Bachman, P., and Suleman, K. (2017). NewsQA: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200, Vancouver, Canada, August. Association for Computational Linguistics.

Weinberger, K. Q., Dasgupta, A., Langford, J., Smola, A., and Attenberg, J. (2009). Feature hashing for large scale multitask learning. In *ICML*.

Zhu, Y., Kiros, R., Zemel, R. S., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 19–27.