Open-domain Question Answering using Wikipedia

Duy Ngoc Nguyen
Computer Science Dept
Simon Fraser University
ndnguyen@sfu.ca

Viet Anh Duong
Computer Science Dept
Simon Fraser University
anhd@sfu.ca

Abstract

In Natural Language Processing (NLP), Open-domain Question Answering (OpenQA) is an intriguing task that can test machine comprehension. OpenQA aims to answer a question in natural language using based on a large-scale document base (Figure 1). Most modern OpenQA researches follow the architecture of "Retriever-Reader" (Zhu et al., 2021), which retrieves relevant document from a knowledge base and extract the answer within. As a knowledge source, Wikipedia has been widely considered to be one of the biggest open-source general encyclopedia with over 6 million articles of various topics (Hewlett et al., 2016). Furthermore, with the release of transformer-based BERT in 2018, the average performance of extractive Question Answering models on the Stanford Question Answering Dataset 2.0 (SQuAD 2.0) (Rajpurkar et al., 2018) made a huge leap, being able to beat human performance level. We aim to build an effective OpenQA system, which we called LookupQA, using Wikipedia as our knowledge base and a transformer-based answer-extraction model.

1 Introduction

1.1 Problem statement

Our project's goal is to build a Retriever-Reader system LookupQA that can output an answer given the user's question prompt. For our system, we need to be able to retrieve the right document relating to the query, and fine-tune an answer-extraction model to output the most relevant answer within the text we found.

1.2 Motivation

Among the downstream tasks of Natural Language Processing (NLP), Question Answering demands a high level understanding of the text as well as the ability to reason to be able to produce a correct answer. In tackling this problem, we hope to evaluate machine comprehension on large scale data. Being able to retrieve specific natural language query from a large knowledge database can simulate a human's ability to reason an answer to a question from memory, which aids in the development of human-like intelligence in machines.

Since the use of transformer-based word representation models is prevalent among QA state of the art and within the material of the course, we decided to do a review of their performance in QA.

2 Related work

2.1 Retriever-Reader model

Chen et al. (2017) introduced the Retriever-Reader model for OpenQA, which used a separate module for information retrieval and answer extraction. Zhu et al. (2021) summarized this approach, which we use as a guideline for our approach. This modular approach allows us to leverage many of the existing researches done in both areas.

Whereas Chen et al. (2017) utilized a recurrent neural network (RNN) for answer extraction, we decided to opt for the transformer-based approach which has been shown to dominate the SQuAD leaderboard in recent years.

2.2 BERT-based answer extraction

For our answer extraction, we examine two models, BERT (Devlin et al., 2018) and AlBERT (Lan et al., 2019).

Devlin et al. (2018) was the first to show tremendous potential of the transformer-based approach. Lan et al. (2019) introduced parameter reduction techniques that resulted in improved training time and performance.

2.3 Datasets

For our fine-tuning dataset for answer extraction, we choose a combined dataset of Rajpurkar et al.

Figure 1: Example on the structure an OpenQA system in action (Zhu et al., 2021)

Dataset	Example	Paragraph
SQuAD 2.0	Q: "The traveling salesman problem is an example of what type of problem?" - A: "A function problem"	Paragraph: "A function problem is a computational problem where a single output (of a total function) is expected for every input, but the output is more complex than that of a decision problem, that is, it isn't just yes or no. Notable examples include the traveling salesman problem and the integer factorization problem."
Adversarial QA	Q: "How does growth hormone affect our body?" - A: "regulate the immune system"	"Hormones can act as immunomodulators, altering the sensitivity of the immune system. For example, female sex hormones are known immunostimulators of both adaptive and innate immune responses. Some autoimmune diseases such as lupus erythematosus strike women preferentially, and their onset often coincides with puberty. By contrast, male sex hormones such as testosterone seem to be immunosuppressive. Other hormones appear to regulate the immune system as well, most notably prolactin, growth hormone and vitamin D."

Table 1: Example on the structure of SQuAD 2.0 and Adversarial QA datasets

(2018) and Bartolo et al. (2020).

Rajpurkar et al. (2018) provided the most widely used benchmark for the natural Question Answering task. Bartolo et al. (2020) created a dataset consisting of adversarial examples that even state-of-the-art models were unable to answer correctly. Bartolo et al. (2020) also found that using AdversarialQA as training set results in better generalization of the model.

3 Approach

In the following we describe out LookupQA system which follows the Retriever-Reader approach (Zhu et al., 2021), consisting of two components: (1) the Document Retriever finding the most relevant article and (2) the Document Reader that comprehends and extracts answer from a collection of documents (Figure 2).

3.1 Document Retriever

Our document retriever utilizes a simple Wikipedia API (wik) to search for the relevant document. We simply pass the whole question string as the search query. Out of the returned list of article titles, we take the top results and get the article full text. These articles are then processed by the Document Reader.

Our retriever was designed with a hyperparameter n as the number of relevant article to return. We initially wanted to manually tune this hyperparameter to find the optimal n. However, due to the network limitation of the API (which we will discuss further in later section), we were only able to obtain the benchmark for our full validation set using $n = 1$.

Note that in practice, the network overhead of obtaining multiple articles is negligible for a small

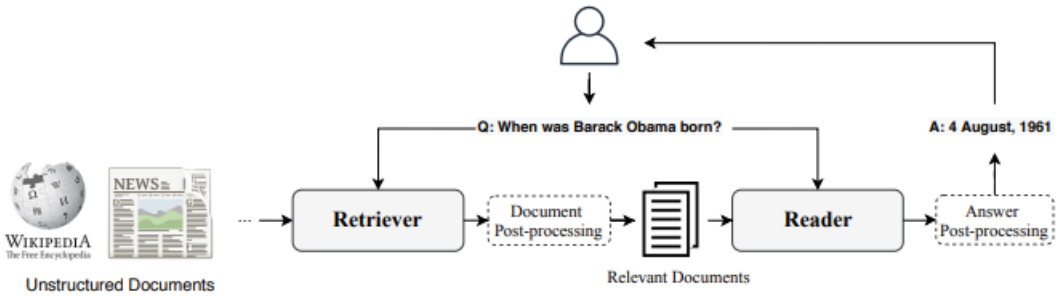

Figure 2: Structure of a Retriever-Reader OpenQA system (Zhu et al., 2021)

collection of queries.

3.2 Document Reader

Our choice of a transformer-based model for Document Reader is inspired by the positive results obtained by these models on machine comprehension tasks. Most notably, BERT-based single and ensemble models dominates the majority of the SQuAD 2.0 leaderboard (Rajpurkar et al., 2018).

We obtain pre-trained base sized BERT, AlBERT models from Huggingface (hug). We then fine-tune them and evaluate them on the joint dataset of SQuAD 2.0 and AdversarialQA.

We preprocess the input by mapping the context start and end positions to the token positions space. Alternatively, if no answer can be found, or the answer is not within the context given, we mark set $(context\_token\_start, context\_token\_end) = (0, 0)$.

We finetune both models with the same training arguments: fixed learning rate $= 2e - 5$, batch size $= 20$, number of epochs $= 3$, weight decay $= 0.01$.

To get the answer, we first deploy a chunker that split the context input into chunks of context that fits into our models' input size of 384. We run the model independently on each chunk of context to obtain the best answer for each chunk, then return the answer with the highest score among all chunks. Additionally, we manually tune our answer threshold $= 0.1$, which categorizes the question as unanswerable with the given context if the score for the best answer falls below the threshold.

3.3 Data

The dataset we're using for fine-tuning and evaluation is combined SQuAD 2.0 (Rajpurkar et al., 2018) and AdversarialQA (Bartolo et al., 2020).

SQuAD 2.0 consists of 150,000 question and answer pairs (Rajpurkar et al., 2018). Each example is composed of a paragraph of context and a human curated question and answer. The answer is always a span found within the given context paragraph. Additionally, 50,000 of examples in the dataset has no answer, which the model has to recognize in order to perform well.

AdversarialQA consists of 36,000 question and answer pairs (Bartolo et al., 2020). Similar to SQuAD 2.0, each examples are human-curated questions and answers based on a context paragraph. Uniquely, the examples found in AdversarialQA consists of questions where popular QA models such as RoBERTa, BERT, and BiDAF fails to give the correct answer.

Table 1 consist of an example from both datasets.

For fine-tuning, we use a combination of 30,000 training examples from AdversarialQA and 130,319 training examples from SQuAD 2.0. For evaluation of our reader performance, we use the dev set of 3,000 examples from AdversarialQA and 11,873 examples from SQuAD 2.0, where we use the context given in the dataset to generate answers. For evaluation of our full model performance, we use the same dev set without the given context, and use the documents returned from our document retriever to extract answers.

3.4 Metrics

Our evaluation metric is (macro-averaged) $F1$ score and EM exact match score, as described by Rajpurkar et al. (2018):

- **Exact match** measures the percentage of predictions that match a ground truth answer exactly (Rajpurkar et al., 2018)
- **(Macro-averaged) F1 score** measures the average overlap between the prediction and ground truth answer. The final score was averaged over all questions (Rajpurkar et al., 2018).

	SQuAD 2.0		AdversarialQA		Combined	
Methods	EM	F1	EM	F1	EM	F1
BERT Document Reader	69.3	72.4	28.4	35.5	61.1	64.9
ALBERT Document Reader	75.5	78.7	29.7	38.1	66.2	70.5
BERT fine-tuned on SQuAD 2.0	69.8	73.1	12.6	18.2	58.3	61.9
DrQA	69.5	78.8	n/a	n/a	n/a	n/a

Table 2: Evaluation results on the SQuAD 2.0, Adversarial QA, and Combined datasets of our models, in compared with pre-trained BERT fine-tuned on SQuAD 2.0 and DrQA (Danqi et al., 2017)

Methods	SQuAD 2.0
DrQA	29.8
BERT LuQA 1-full	20.8
ALBERT LuQA 1-full	20.6

Table 3: Full model results on the SQuAD 2.0 dataset, in compared with DrQA (Danqi et al., 2017)

4 Experiments

We first present the evaluations of our Document Reader separately, then the full model LookupQA performance.

4.1 Evaluation of Document Reader

We first examine the performance of our Document Reader on 3 datasets: SQuAD 2.0, AdversarialQA, and the joint dataset from both.

We use a baseline model of base sized BERT finetuned only on SQuAD 2.0 for our comparison.

Result and analysis: Table 2 presents our evaluation results on each of the datasets. We found that finetuning on the joint dataset only results in negligible performance decrease on the SQuAD 2.0 metrics, while significantly boosts the performance on the joint dev set metrics. Comparing between the two versions of our Document Readers, we find that AlBERT significantly out performs BERT in all of the metrics we gathered. The performance gap between our

4.2 Full System Question Answering

We assess the performance of our full system LookupQA on the SQuAD 2.0 dataset.

Our LookupQA results are obtained with Document Retriever returning 1 best match document.

We use the results of DrQA (Chen et al., 2017) as comparison baseline

Result and analysis: Table 3 presents our evaluation results. We found that our model significantly under-performs compared to DrQA. Upon further analysis, we attribute this lackluster performance to: (1) Worse Document Retriever performance and (2) Valid answers that's different from gold answers

- (1): Due to the networking nature of the Wikipedia API we use, when evaluating on the 11,873 examples on SQuAD 2.0 dev set, we are only able to get a maximum of one document per query before the total API overhead becomes too long and the API server becomes unresponsive. In contrast, DrQA's approach to their retriever model can return 5 relevant documents, thus increasing their chance of finding SQuAD's gold answer.

- (2): Table 4 presents an example where our model returns a sensible and valid answer that differs completely from the SQuAD 2.0's gold answer. This simply mean that the some labeled 'wrong' answers from our system aren't necessarily wrong but reflects the difference in the scope of the context that was gathered by SQuAD and the context gathered by Document Retriever.

5 Limitations

We identify several limitations to our approach and possible improvements to them.

5.1 Document Retriever: Wikipedia API

While using the Wikipedia API saves us considerable amount of time and storage resources, being

dependent on the API for each prediction also results in considerable network overhead from the API. The API overhead scales with the number of articles we want to return, making obtaining predictions with more than 1 returned article extremely time-consuming and prone to API failures. Due to this, we are also unable to conduct experiments that compares different number of documents returned and how they affect the model performance.

For future improvements, we want to investigate and implement more complex retriever models outlined by Zhu et al. (2021), which is able to not only help with prediction time but also prediction accuracy.

5.2 Evaluation of full model

Partially due to the limitations of our Document Retriever outlined above, we were unable to obtain additional benchmarks on Wikipedia specific OpenQA datasets such as Hewlett et al. (2016), which we believe to be more indicative of the model's real world performance then the SQuAD 2.0 metrics as the gold answer is obtained from the same contexts as our model.

6 Conclusion

From this project, we studied the task of machine comprehension in a larger scale by obtaining knowledge from Wikipedia for open-domain question answering. Our approach combines Wikipedia querying API with finetuning transformer models using adversarial examples. Our results confirms the use of adversarial examples leading to better generalization performance. However, it also points out the flaws of relying on a networking API, which is infeasible for large amount of queries.

Furture work should aim to improve upon our LookupQA system. We would like to (1) implement a better Document Retriever component that uses local Wikipedia dumps and (2) finetuning Document Reader on datasets with longer context articles, rather than short context snippets.

7 Contributions

Our roles are divided into the following:

- Duy Nguyen: write documents, created the dataset by combining two existing ones, implemented Document Reader and full LookupQA pipeline, design experiments and gather baseline results, analyze experiment results.
- Viet Anh Duong: wrote documents and report, trained ALBERT models, implemented Document Retriever, helped with testing and conducting experiments, contributed in analysing experimental results.

References

Hugging face – the ai community building the future.

Wikipedia¶.

Max Bartolo, Alastair Roberts, Johannes Welbl, Sebastian Riedel, and Pontus Stenetorp. 2020. Beat the AI: investigating adversarial human annotations for reading comprehension. *CoRR*, abs/2002.00293.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. *CoRR*, abs/1704.00051.

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

Daniel Hewlett, Alexandre Lacoste, Llion Jones, Illia Polosukhin, Andrew Fandrianto, Jay Han, Matthew Kelcey, and David Berthelot. 2016. WIKIREADING: A novel large-scale language understanding task over Wikipedia. In *Proceedings of the The 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations.

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

Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. 2021. Retrieving and reading: A comprehensive survey on open-domain question answering. *CoRR*, abs/2101.00774.