Mindstone: DeepIR with contextualized attention-based network for open-domain QA

Madhulima Pandey
Stanford University
mpandey8@stanford.edu

Abstract

This project aims at improving how machines can answer questions directly from large collections of documents. We have proposed a new pipelined approach for open-domain question answers that improves on DrQA (Chen et al., 2017). To achieve that goal we have introduced two new models - DeepIR and FDreader. The DeepIR model repurposes BERT embedding with other learned parameters for precomputing and ranking documents for the retrieval task. FDreader model uses BERT pre-trained contextualized embedding, and paragraph ranking for predicting answer spans. We show that the pipeline models produces solid results on the SQuAD and Wikipedia dump datasets - DeepIR with its neural search technique performs significantly better than the DrQA doc retrieval results. We report FDreader model improves the accuracy of Doc Reader by more than 10 points.

1 Introduction

Machines have the potential to improve their answering capability by using the vast amount of information available online as in Wikipedia, blogs, forums, websites, social media posts for their knowledge sources. However, using this dynamic and rich information is a hard problem as programs cannot exploit any ontology and instead have to search a large collection of documents and find the relevant documents for answering the question. Chen et. al (2017) proposed the DrQA system to tackle this problem through machine reading at scale. The DrQA system has a few limitations. Its retriever uses n-gram hash based matching for document retrieval, which while fast, could miss out on semantically matching documents. Furthermore, the reader works on individual paragraphs (in k documents) to answer questions, and does not work at the document-level. The reader performance drops from 70% for paragraphs to 50% over the full document (Raison et al., 2018). In this work, we focus on improving both the reader and the retriever so that the system is more robust to longer contexts, retrieves the more relevant document and hence is more accurate overall.

2 Related Work

The machine reading task of learning to automatically answer questions given a piece of text has been making significant progress in recent years. There are two primary trends driving this progress - the first is the creation of training and evaluation datasets like QACNN/DailyMail based on news article (Herman et. al., 2015), CBT based on children books (Hill et al. 2016), WikiQA (Yang et al., 2015) and SQuAD (Rajpurkar et al., 2016) based on Wikipedia, or QAngaroo based on Wiki and medical publications (Welbl et al. 2017). The second is the recent advances in deep-learning architectures like attention-based and memory augmented neural networks (Bahdanau et al. 2015; Weston et al., 2015), Transformers (Vaswani et al., 2017) and Multitask learning (McCann et al., 2018) that can boost the machine comprehension of texts.

Until recently, open-domain question answering has been mostly addressed through the task of answering from structure knowledge bases such as WebQuestions (Berant et al., 2013) and SimpleQuestions (Bordes et al., 2015). However, limitations of KBs such as incomplete or missing information, fixed schemas, etc. and the recent advances in the machine reading have allowed new progress in the field of question answering from a large corpus of unstructured documents. We see new datasets for attacking open-domain QA problem such as MSMARCO(Nguyen et al., 2016) where questions sampled from real anonymized user queries are paired with real web documents from the Bing search engine, TriviaQA (Joshi et al., 2017) that includes question answer pairs authored by trivia enthusiasts along with independently gathered evidence, and the most recent dataset Natural Questions (Kwiatkowski et al., 2019) where real anonymized Google search queries are paired with human annotated answers from Wikipedia pages for two types of responses: long answers and short answers.

DrQa system tackles the open-domain question answering through machine reading at scale, that is answering questions using spans of tokens extracted from Wikipedia. The DrQA pipeline combined a document retriever and a document reader which integrates search with the multi-layer recurrent neural network. There have been numerous work that has proposed to tackle SQuAD dataset only (Dhingra et al., 2016; Wang et al., 2017b; Xiong et al., 2017; Hu et al., 2017) but only a few pipeline approaches that have attempted to deal with selecting answers from multiple documents with multiple paragraphs in them such as Reinforced Mnemonic Reader (Hu et al., 2017) and Reinforced Reader-Ranker (Wang et al., 2017a).

3 Approach

The Mindstone system for machine reading at scale consists of a multi-stage pipeline (Figure 1. The first module, **DeepIR**, is a document retriever and consists of two stages. The first stage retrieves n document from the corpus in response to a user question q. The second stage is the document ranking system that ranks these n documents and identifies the top k (typically k=5) documents that are likely to contain the answer. The second module **FDreader** selects the paragraph within the one or several documents using the un-normalized and un-exponentiated score and returns the correct answer span.

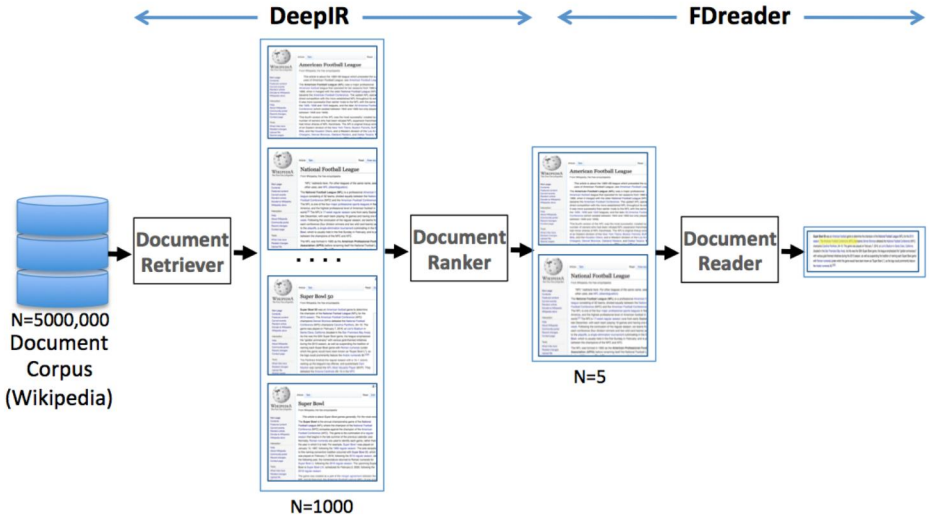

Figure 1: MindStone: Machine Reading at Scale

3.1 DeepIR - Document Retriever

Fast and efficient search over large document corpus needs precomputation of document features that can be efficiently matched with search query at run-time. TF-IDF based approaches are fast, but could miss out on semantically matching documents. Neural network with deep learning provides better accuracy in identifying relevant documents but are computationally intensive. We made an earlier attempt to build a full neural network for retrieval but lacked compute resources and time to

train. (Appendix A describes the results and our analysis of this approach). Hence, we have built a document retrieval system that combines the strengths of both approaches.

The challenge of DrQA is illustrated in Figure 2.

Num of docs	Question	Answer	Wikipedia Article with the max score
5	"What is question answering"?	*A computer science discipline within the fields of information retrieval and natural language processing*	Question answering
100	"What is question answering"?	*the contestant wins a certain amount of money*	Who Wants to Be a Millionaire?

Figure 2: DrQA Reading errors as n_docs increases from 5 to 100.

The results show that the response to the first question on the DrQA github: "What is question answering?", with n_docs = 5. The correct answer is returned. However, as n_docs is increased to 100, even the top 5 answers returned by the reader are not correct. For SQuAD questions, the accuracy of Doc Retriever for n_docs = 5 is 78% and with n_docs = 100 the accuracy increase to 92%. However, as we increase the accuracy of Doc Retriever by increasing n_docs, the accuracy of the reader drops significantly e.g., for 100 articles, i.e., 6000 paragraphs, the reader performance can drop by more than 20% Therefore, it is necessary to further narrow the search space using semantic matching methods so that we can increase the accuracy of retriever without penalizing the reader.

The first stage of **DeepIR** compares articles and questions using TF-IDF and bigram hashing using DrQA modules. This stage returns the best *n* articles. In our model the default value of n is 100.

In the second stage these *n* documents are scored and ranked by a BERT based neural network, and the top *k* documents are retrieved and then sent to the **FDreader** module. The job of the ranker is to estimate a score s_i of how relevant a document d_i is to a query q. The ranker model architecture is based on fine-tuned BERT model with precomputed contextual embeddings with an additional dropout and linear layer (Figure 3) (Nogueira et al., 2019). The query q is fed as segment A, and the document d_i is fed as segment B. The max sequence length of q is 16 words, and the max sequence length of d_i is 500 words, such that the sum of the length of q, d_i and special tokens do not exceed the BERT model limitation of 512 tokens. We preprocess the document such that d_i consists of tokens from the start of the document, ignoring section titles, and paragraphs with less than 150 characters to reduce noise in the dataset.

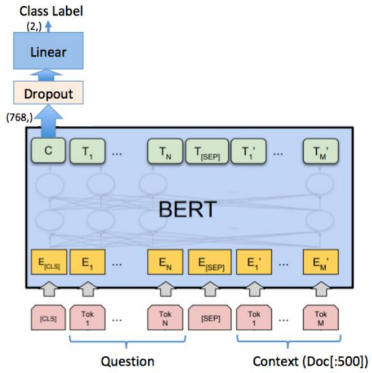

Figure 3: Document Ranking

The score for each of the *n* documents is computed independently and then the model predicts the best top *k* documents. We use a pre-trained BERT model and fine-tune it for document ranking task using the cross-entropy loss:

$$Loss_{ranker} = -\sum_{j \in J_{pos}} log(d_j) - \sum_{j \in J_{neg}} log(1 - d_j) \quad (1)$$

where J_{pos} and J_{neg} are the indices of the relevant and non-relevant articles, respectively for a query. We train using a dataset derived from the SQuAD train examples, and measure performance of the document retrieval module.

3.2 FDreader - Document Reader

In Figure 4, below, we illustrate the approach for question answering, based on the precomputed embeddings and additional linear layer for predicting the span of tokens that is most likely the correct answer.

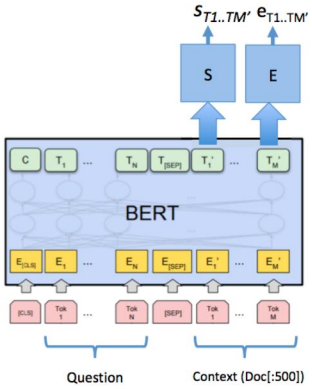

Figure 4: Question Answering

We use Bert (Devlin et al., 2018) base model, and we represent the input question and paragraph as an input sequence, with the segment id A for the question, and the segment id B for the paragraph. The additional parameters used are $S \in \mathbb{R}^h$, and $E \in \mathbb{R}^h$, where $h = 768$ for Bert-base. As described in [Bert-devlin], for each token i in the paragraph, let T_i be the hidden state. The probability $P_{start}(i)$ of token i being the start of the answer span is computed as a dot product between T_i and S followed by a softmax over all of the tokens in the paragraph. Similarly the probability of end of answer span, $P_{end}(i)$ is computed as a dot product between T_i and E followed by softmax.

4 Dataset

From the SQuAD dataset and a database of wikipedia dump, consisting of over 5 million articles, we construct the dataset for DeepIR (Figure 5).

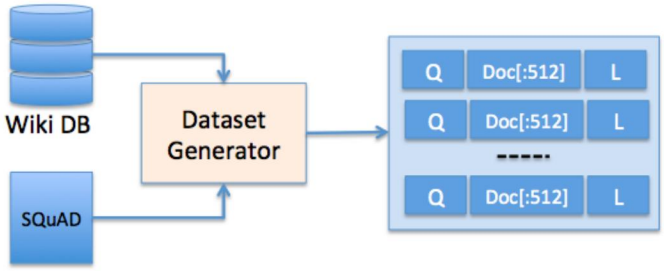

Figure 5: Dataset generation from SQuAD and Wikipedia article database

We construct a set of Question-Article-Label triples (Figure 6, where the Question is from SQuAD, the Article is the entire article from Wikipedia text (which includes the corresponding SQuAD context, i.e., para from the Wikipedia article), and the Label is a binary value that indicates if the article contains the paragraph that answers the question. For negative examples, we randomly select a non-matching article for the question. This dataset is created using the Wikipedia text database available as part of DrQA (Chen et. al (2017), and the SQuAD 1.1 dataset. Our choice of SQuAD 1.1, over 2.0 is dictated by the availability of the corresponding Wikipedia dump from available from DrQA github. We use the approach of (Devlin et al., 2018), to tokenize the examples using a 30,522 wordpiece vocabulary, limiting the total size of question and article in each instance to 512 tokens as the BERT models do not handle large documents with thousands of words.

Question	Document	Label
Which NFL team represented the AFC at Super Bowl 50?	Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season. The American Football Conference (AFC) champion Denver Broncos defeated the National Football Conference (NFC) champion Carolina Panthers 24–10 to win their third Super Bowl championship. The game was played on February 7, 2016, at Levi's Stadium in Santa Clara, California (located in the San Francisco Bay Area). As this was the 50th Super Bowl, the league emphasized the "golden anniversary" [... 512 words]	1
Which NFL team represented the AFC at Super Bowl 50?	Kenya (;), officially the Republic of Kenya, is a country in Africa and a founding member of the East African Community (EAC). Its capital and largest city is Nairobi. Kenya's territory lies on the equator and overlies the East African Rift covering a diverse and expansive terrain that extends roughly from Lake Victoria to Lake Turkana (formerly called Lake Rudolf) and further south-east to the Indian Ocean. It is bordered by Tanzania to the south and southwest, Uganda to the west, South Sudan to the north-west, Ethiopia to the north and Somalia to the north-east. Kenya covers , and had a population of approximately 45 million [...512 words]	0

Figure 6: Question Answering

During the creation of this dataset, we discarded the questions for those articles for which we could not find a matching wikipedia article, even after accounting for UTF character normalization and underscore to space replacment in article IDs (titles) in our database searches. Our final dataset for DeepIR consists of 169,948 training and 20,924 dev examples. For FDreader question answering, we use the SQuAD 1.1 dataset for consistency with the retriever.

5 Experiments and Results

5.1 Experiments

Our first set of experiments were with DeepIR. We use a BERT base model with 12 layers, 110M parameters and additional dropout and linear layer. We used Adam optimizer with a learning rate of $5.0 \cdot 10^{-5}$, with a warmup proportion of 0.1, i.e., the proportion of training to perform linear rate warmup,beta1 = 0.9, beta2 = 0.999, L2 weight decay of 0.01. We use a dropout probability of 0.1 on all layers. We used a batch size of 4 for training with sequence length (question + document size) of 512, and batch size of 12 for sequence length of 256. A batch size of 16 is used for evaluation.

We evaluated the performance of DeepIR by determining for (question, context) pair in SQuAD, the probability of retrieving a relevant document (i.e., which contains the context) among the top k articles $(k = 5)$, given the question. The overall retriever performance was measured by retrieving $n = 100$ articles using TFIDF and bigram matching, and from these, then selecting the top ranked $(k = 5)$ artices using the second stage of DeepIR.

Performance and latency is an important meaure for reading at scale problems, and we have included a number of measurements in the next section.

For FDreader, we have used similar hyperparameter settings, with Adam optimizer learning rate of $5.0 \cdot 10^{-5}$, warmup proportion of 0.1, beta1 = 0.9, beta2 = 0.999, L2 weight decay = 0.01 and a dropout probability of 0.1. Our measures for performance here are the F1 and EM scores on SQuAD.

These experiments were done on a RTX 2080ti, with a i9-9900X system, which has about twice the floating point performance of a standard K-80 gpu cloud instance.

Our code is built on components from DrQA and pytorch BERT. The github are listed as the last two references.

	Document Size	Batch Size	Num. Epochs	Eval Accuracy	Runtime
Model 1	256	12	1	95.5 %	1h:53m
Model 2	512	4	1	96.1 %	3h:23m
Model 2	512	4	5	83.4 %	14h:59m

Table 1: Training DeepIR

Retrieval Technique	Retrieval Accuracy	Retrieval Time/query
TF-IDF(k=1)	36.9%	47.1ms
TF-IDF(top 100) + Ranking(k=1)	50.3%	1.779secs
TF-IDF(k=5)	77.9%	47.0ms
TF-IDF(top 100) + Ranking(k=5)	85.2%	1.807secs

Table 2: DeepIR Performance

5.2 Results

In table 1, we show the results of the training the retreiver. We noticed that the fine-tuned models quickly saturate in eval accuracy within a small number of epochs and larger sequence sizes do not necessarily yield better results.

We established the baseline from DrQA after multiple code updates to it to bring it to the current version of PyTorch, and updated the tokenizer. We ran experiments with Python 3.5, PyTorch 1.0.0, and CoreNLP 3.9.1 (all other CoreNLP versions do not work with DrQA). With k = 5, we measured the retrieval accuracy of DrQA to 78%. and with k=100, the accuracy rose to 92%.

We created the DeepIR model and trained in on a small part of the dataset described above with 179K train, and 18K dev examples, and measured the retrieval accuracy of 93%. In the full retrieval pipeline, the over retrieval accuracy with fetching 100 documents, and then selecting $k = 5$ using our model, gave an accuracy of 85.2%, a gain of over 7 points over DrQA, for $k = 5$. The accuracy numbers are included in Table 2.

We have also included the retrieval times in this table. The TF-IDF-based approach takes a about 47 milliseconds to do the retrieval across the entire Wikipedia dataset. The reranker task then takes about 17 milliseconds for scoring each retrieved document and question pair. While there is score for more optimization in our implementation (e.g., precomputing tokens will reduce the time by 25-30%), the overall response latency is proportional to the number of documents that are ranked.

For the reader task, we established the reader baseline from DrQA. For SQuAD, the F1 and EM scores are 68.49% and 77.83% , This takes 369 minutes of run on a nv6 azure data science VM instance. Our FDreader implementation has F1 and EM scores of 80.1% and 87.9%, respectively (Table 3).

5.3 Analysis

We have achieved improved retrieval results by incorporating a neural document ranking approach. While the run-time of retrieval increases in proportion to the number of question-document pairs that are ranked, the accuracy increases by 8-10 points. However, when we closely looked at some of the retrieval errors, our conclusion was that SQuAD questions may not be best suited for evaluating Open Domain QA systems. For example, given the question 'When were the finalists announced?', the reference article is 'Super Bowl 50', but the retriever identifies 'American Idol', 'Beyonce', 'Premier League' among the top 5 matches. These are all very plausible answers, as they contain

Reader	F1	EM
DrQA Document Reader	69.5%	78.8%
FDreader	80.1%	87.9%

Table 3: DeepIR Performance

information on the announcement of finalists. So, the fact that reference article did not show up among the top-k results is suggestive of the need for a different dataset of questions. Additionally, while our reader selects the paragraph within the one or several documents using the un-normalized and un-exponentiated score and returns the correct answer span, there is scope for considerable improvement by taking into account features such as paragraph position in the document, TF-IDF vector matches at the paragraph level, token counts preceding the paragraph. We will be continuing some of these enhancements beyond this course.

We have tried additional experiments for encoding documents using pre-trained embeddings using a model described in Appendix A, (Figures 7, 8), and in our milestone report. The motivation was to use the pre-computed document encodings to enable efficient retrieval of matching documents. However, our initial attempts did not yield more than 75% accuracy. On examining the failures, and the BERT models, we have come to the conclusion that these models have been trained on specific tasks (masked language model and next sentence prediction), which makes them work for many language tasks. However, the task of encoding generation is of a different natures, requires additional model elements, such as an attention layer between the q and p encodings, and also all of the hidden state corresponding to the question and paragraph tokens should be used in the model, not just the hidden state correspoding to [CLS] token.

6 Conclusion and Future Work

In this project we introduced Mindstone, a pipeline system with DeepIR retrieval and FDreader machine reader. Our experiments have shown that this pipeline, a combination of deep learning-based information retrieval with document reading, significantly improves the performance of Open Domain question answering.

Future work entails improving the full-document reading including multi-paragraph handling in the context of the document, rather than to reading paragraphs individually. The pipeline has been designed as a general architecture that is capable of running on other datasets, such as NQ and TriviaQA, as currently we were limited by time constraints. We also plan to continue work on the neural retriever approach to improve its efficiency and task performance using precomputed document encodings.

7 Additional Information

Mentor: Prof. Chris Manning

References

[1]Chen, Danqi, et al. "Reading Wikipedia to answer open-domain questions." arXiv preprint arXiv:1704.00051 (2017).

[2]Zhu, Chenguang, Michael Zeng, and Xuedong Huang. "SDNet: Contextualized Attention-based Deep Network for Conversational Question Answering." arXiv preprint arXiv:1812.03593 (2018).

[3]Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

[4]Rajpurkar, Pranav, Robin Jia, and Percy Liang. "Know What You Don't Know: Unanswerable Questions for SQuAD." arXiv preprint arXiv:1806.03822 (2018).

[5]Raison, Martin, et al. "Weaver: Deep co-encoding of questions and documents for machine reading." arXiv preprint arXiv:1804.10490 (2018).

[6]Nogueira, Rodrigo, and Kyunghyun Cho. "Passage Re-ranking with BERT." arXiv preprint arXiv:1901.04085 (2019).

[7]Pennington, Jeffrey, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation." Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014.

[8]Clark, Christopher, and Matt Gardner. "Simple and effective multi-paragraph reading comprehension." arXiv preprint arXiv:1710.10723 (2017).

[9]Dai, Zihang, et al. "Transformer-XL: Language Modeling with Longer-Term Dependency." (2018).

[10]Alberti, Chris, Kenton Lee, and Michael Collins. "A BERT Baseline for the Natural Questions." arXiv preprint arXiv:1901.08634 (2019).

[11]Seo, Minjoon, et al. "Bidirectional attention flow for machine comprehension." arXiv preprint arXiv:1611.01603 (2016).

[12]Wang, Shuohang, Yu, Mo, Guo, Xiaoxiao, Wang, Zhiguo, Klinger, Tim, Zhang, Wei, Chang, Shiyu, Tesauro, Gerald, Zhou, Bowen, and Jiang, Jing. R3: Reinforced reader-ranker for open-domain question answering. arXiv preprint arXiv:1709.00023 (2017a).

[13]Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).

[14]Weston, Jason, Sumit Chopra, and Antoine Bordes. "Memory networks." arXiv preprint arXiv:1410.3916 (2014).

[15]Hermann, Karl Moritz, et al. "Teaching machines to read and comprehend." Advances in Neural Information Processing Systems. 2015.

[16]McCann, Bryan, et al. "The natural language decathlon: Multitask learning as question answering." arXiv preprint arXiv:1806.08730 (2018).

[17]Joshi, Mandar, et al. "Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension." arXiv preprint arXiv:1705.03551 (2017).

[18]Bordes, Antoine, et al. "Large-scale simple question answering with memory networks." arXiv preprint arXiv:1506.02075 (2015).

[19]Berant, Jonathan, et al. "Semantic parsing on freebase from question-answer pairs." Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. 2013. [20]Kwiatkowski, Tom, et al. "Natural Questions: a Benchmark for Question Answering Research." (2019).

[21]Nogueira, Rodrigo, and Kyunghyun Cho. "Passage Re-ranking with BERT." arXiv preprint arXiv:1901.04085 (2019).

[22] https://github.com/facebookresearch/DrQA

[23] https://github.com/huggingface/pytorch-pretrained-BERT

8 Appendix

8.1 DeepIR - Document Retriever

8.1.1 Model

We have developed DeepIR, a new model for retrieving relevant documents for answering the question. The goal is to determine if a document d is relevant to answering a question q. The documents can be extremely large – articles with 1000's of words in Wikipedia are common. So, we simplify this to the problem into a more tractable one, that of determining the relevancy of a paragraph to a question, and then take the relevancy of individual paragraphs as an indication of the relevancy of the document.

We define the relevancy as a classification problem of determining if a paragraph p is relevant to answering a question q. As described ahead, we create a model for determining this relevancy and precompute encodings for all paragraphs for all documents in the corpus. During retrieval time, given a question q, the relevancy score of each document is the score of the paragraph within it with the maximum relevancy score pertaining to that question. The top k documents are returned to the document reader, and as in DrQA, we have a default value of k=5.

Formally, the training set is defined as a 3-tuple, (q, p, l), where q is question, p is a paragraph and label $l \in \{0, 1\}$ indicates relevancy of p to q. Each question and paragraph is converted into a sequence of token ids of lengths 32, and 256 respectively, based on a 30522 wordpiece vocabulary. The sequence ids and mask values are set as described in [2] for single sequence problems. As in figure 1, we use two separate BERT (base model) pretrained embedding modules, PCE_Q, and PCE_Doc, to generate embeddings $q_{emb} \in \mathbb{R}^{768}$ and $p_{emb} \in \mathbb{R}^{768}$ of p and q respectively.

Using a linear projection, dropout and tanh nonlinearlity, we generate the encodings for p and q.

$$q_{encoding} = tanh(Dropout(\mathbf{W_q} q_{emb} + b_q)) \qquad q_{encoding} \in \mathbb{R}^{768} \tag{2}$$

$$p_{encoding} = tanh(Dropout(\mathbf{W_p} p_{emb} + b_p)) \qquad p_{encoding} \in \mathbb{R}^{768} \tag{3}$$

Finally, we multiply element-wise $q_{encoding}$ and $p_{encoding}$ and pass it through a projection and sigmoid layer for determining the relevancy.

$$relevancy\_match(q, p) = \sigma(\mathbf{W_{rel}}(q_{encoding} \odot p_{encoding}) + b_{rel}) \qquad W_{rel} \in \mathbb{R}^{1536x1} \tag{4}$$

The model is trained by minimizing the loss function

$$Loss = -CrossEntropy(relevancy\_match(q, p), label(q, p)) \tag{5}$$

For determining whether a document d consisting of paragraphs $\{p_1, p_2, ..., p_N\}$ is relevant to question, we compute $relevancy(q, p_i)$ for all paragraphs in the document and select the maximum value of $relevancy(q, p_i)$ as the relevancy of the document d. There are other alternatives such as maxpooling the paragraph relevancy scores, or higher weighting the scores of earlier paragraphs in the document that we plan to explore.

Once we have the trained model, we use PCE_Doc and the associated projection layer with $\mathbf{W_p}$ to precompute the encodings for the entire document corpus, as in Figure 2. During retrieval, the question encoding is computed using PCE_Q and $\mathbf{W_q}$ and is matched with the encoding of the entire document corpus. This approach is similar to the pre-computed TF-IDF approach of DrQA, but allows for semantic matching to do document retrieval, nut just word or n-gram matchings.

Our approach allows for extremely efficient retrieval across the document as compared to other neural-based approaches such as document-reranking in [6], by precomputing encodings, but using only a set of linear operations with 1536 FP multiply-add and a sigmoid operation.

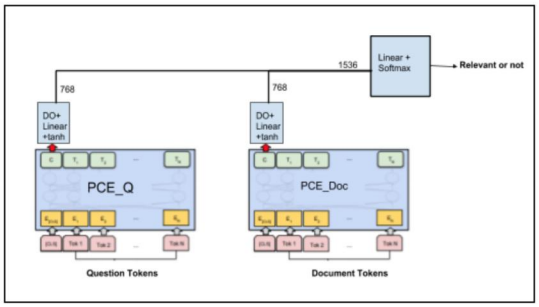

Figure 7: DeepIR model

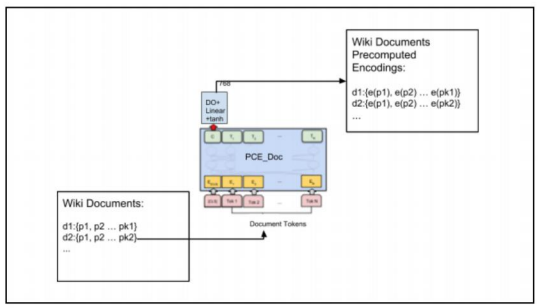

Figure 8: Contextualized Embeddings

8.1.2 Dataset

8.2 FDreader - Document Reader

8.3 Model

Our reader model is based on SDnet [2] (Figure 3) that combines precomputed contextualized embeddings with inter-attention between the question and context. Our model differs from SDnet in two main ways: we allow the BERT model parameters to be fine-tuned during training (SDnet uses locked weights), and we eliminate the self-attention layers in question and context, as that is already captured as a part of the pre-trained embedding blocks.

There are three components in our model - Encoding Layer, Integration Layer, and Output Layer. In the Encoding Layer, the weighted sums of hidden states for all layers in BERT is used to obtain the contextualized embedding. In addition to these contextualized word embeddings for question and context, we also use 300-dim GloVe embedding [7]. For each word, in addition to these embeddings, we have a feature vector that included POS embedding, NER embedding, an exact matching vector following the approach in DrQA. The Integration Layer employs the following, i) Word-level inter-attention from question to context (passage) based on GloVe word embeddings, ii) two separate BiLSTMs for the contextualized understanding for context and question, iii) multilevel attention from question to context based on all layers of generated representations. The Output layer uses the final question understanding representation and the final representation of context to compute the probabilities of the answer span start and end for context.