Multi-Paragraph Q&A

By

Muhammed Favas K
Ishwar Govind
Katragadda Yagnesh
Joel Sam Mathew
Aishwarya H

August 4, 2023

Contents

1 Introduction **3**

2 Literature Review **3**

3 Methodologies **4**
- 3.1 Paragraph Retriever . . . 4
  - 3.1.1 Pre-Processor . . . 4
  - 3.1.2 Retriever . . . 5
- 3.2 Question Answering . . . 6
  - 3.2.1 Encoder . . . 6
  - 3.2.2 Decoder . . . 7
  - 3.2.3 Internal Verifier . . . 7
  - 3.2.4 Answerability Verification . . . 7
  - 3.2.5 Training Methodology . . . 8
  - 3.2.6 Data Preprocessing . . . 8
  - 3.2.7 Data Augmentation . . . 8

4 Experiments and Results **8**
- 4.1 Paragraph Retrieval . . . 8
  - 4.1.1 Retrieval Method Selection . . . 8
  - 4.1.2 Selection of parameters . . . 8
  - 4.1.3 Model Selection . . . 9
- 4.2 Question Answering . . . 9
  - 4.2.1 Model Selection . . . 9
  - 4.2.2 Selection of Max Length of Input Sequence . . . 9
  - 4.2.3 Finding Non-Answerability Threshold . . . 10
  - 4.2.4 External Verifier . . . 10
  - 4.2.5 Fine Tuning on New Dataset . . . 10

5 Conclusion **10**

1 Introduction

The ability of a machine to read and comprehend a text similar to a human is known as machine reading comprehension (MRC). It entails being able to comprehend the text's main ideas, conclusions, and details as well as being able to respond to inquiries about the text. Natural language processing (NLP), which entails teaching machines to process and comprehend human language, includes MRC as a subfield. Search engines, chatbots, and educational software are just a few of the applications for MRC, an active area of NLP research. In this project, our objective is not only to comprehend a paragraph and comprehend the question we need to retrieve a paragraph from a given set of paragraphs that can further comprehend and answer the question. The main objective of this project is to retrieve a span of text from an appropriate paragraph. We can categorize it as a span-based MRC task. The following sections will give the details of the literature review and methodologies we tried in solving this task.

2 Literature Review

The research on Machine Reading Comprehension is a popular and active task in Natural Language Processing(NLP). Due to the creation of various large-scale benchmark datasets like Stanford Question Answering Dataset (SQuAD)[1], the new version of the SQuAD dataset (SQuAD 2.0)[2] with the inclusion of unanswerable questions, TriviaQA [3], NewsQA [4], and corpora of news with associated queries from the CNN and Daily Mail websites [5] remarkable advancements has been made in this task of machine reading comprehension. Early machine reading comprehension explored the attention-based interaction between the paragraph and question. Some of the major works used attention sum [6], self-matching attention [7], gated attention[8], bi-attention[9] and attention over attention[10]. Recently Pre-trained Language Models(PrLM) were used as the encoder that achieved greater performance on the benchmark datasets. The Pre-trained Language Models are capable of capturing contextualized sentence-level representation which greatly helps the machine reading comprehension. These PrLMs includes ELMo[11], GPT[12], BERT[13], XLNet[14], RoBERTa[15], ALBERT[16], and ELECTRA [17].

The early researchers solved the problem by assuming that every question can be answered from the given set of paragraphs. But in real-world scenarios, this won't be the case where there will be questions that can't be answerable. In such cases, the model returns a null string instead of producing some plausible answers. Very few researchers paid attention to this matter. Some of the studies employed threshold-based answerable verification (TAV) strategy which decides whether the question is answerable or not from the predicted start and end probabilities[7][13][14][15]. Liu et al. [18] used a binary classifier for checking a question is answerable or not. Hu et al.[19] used two types of loss, independent span loss, and independent no-answer loss to better answer extraction and no-answer detection. Further employed an extra verifier to check whether the predicted answer is correct or not. Back et al. [20] extracted a list of conditions from the question and proposed an attention-based loss function for checking the extracted conditions are met by the candidate answer obtained from the MRC model. Zhang et al.[21] proposed a model by inspiring how humans answer a question that integrates two stages of reading and verification strategies. Sketchy reading estimates overall interactions of paragraph and question and gets an initial judgment. Intensive reading that verifies the answer and gives the final prediction.

Ahmad et al[22] proposed neural encoding models and classical information retrieval techniques for effectively answer a question from a long document. Clark et al.[23] proposed a novel method for multi-paragraph reading comprehension by sampling multiple paragraphs during training and using a shared-normalization training objective that encourages the model to produce globally correct output. They combined this method with a state-of-the-art pipeline for training models on document QA data.

Similar to answerability prediction paragraph retrieval is an important task in a question-answering task. Research by Md Moinul Hoque and Paulo Quaresma et al presents a model for efficient paragraph retrieval[24]. The model focuses on the structure and organization of information, analyzes the user's question in depth, and allows for priority-based searching of paragraphs. Its performance was measured in terms of R-precision, Recall, and Mean Average Precision. The results showed that the model was effective and efficient, and could potentially be integrated into real-time QA systems. Though the standard models focused widely on the relevancy of the documents, it is important that the relevancy is measured in real-time. Craig Macdonald and Nicola Tonellotto in their research[25] focused on the usage of Approximate Nearest Neighbor Selection that builds the index structures narrowing down the search space and reducing the latency at the cost of efficiency of the retrieval. An ACL paper in 2019 by Nils Reimers and Iryna Gurevych[26] uses Siamese and triplet network topologies to enhance a pre-trained BERT network and adds a pooling operation to BERT's output to produce a fixed-sized sentence embedding vector. The resulting embedding vector is more suited for comparing the similarity of sentences in a vector space and can provide efficient comparisons for retrieval.

3 Methodologies

We approached the paragraph prediction and question-answering tasks separately. The schematic diagram of the entire system is shown in figure 1. The paragraph retriever module will rank the given set of paragraphs corresponding to a question and retrieve the top two ranked paragraphs. The question and retrieved paragraphs are then fed into the question-answering module which is capable of predicting the span of the answer from the paragraph. The question-answering module then processes each pair of questions and paragraph one at a time and predicts the span probabilities and answerability probability. Based on these probabilities, the answerability verification block will decide which paragraph can give the answer span if the question is answerable. The different methodologies used for the two sub-tasks are explained in upcoming sections.

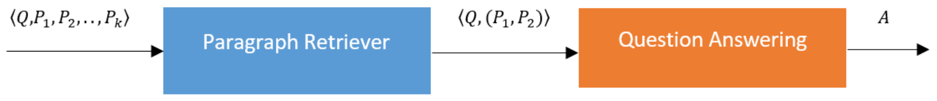

$\langle Q, P_1, P_2, .., P_k \rangle$
Paragraph Retriever
$\langle Q, (P_1, P_2) \rangle$
Question Answering
A

Figure 1: Schematic diagram of the overall system. Paragraph retriever takes a question(Q) and set of paragraphs($P_1, P_2, \ldots, P_k$) and retrieves a paragraph which is further used by the question answering module to predict the span of the answer (A)

3.1 Paragraph Retriever

In this module, the main intent is to identify the paragraph that contains the answer to the given question. It is a retrieval task in which the model takes a query Q and a list of paragraphs $P_1, P_2, ..., P_n$ and predicts the two most likely paragraphs P_1 and P_2 that may contain the answer to Q. The architecture diagram of the proposed system is given in Figure 2.

3.1.1 Pre-Processor

In this task, we begin with the pre-processing of the paragraphs to prevent the impact of noise (non-informative words) on the relevance of the documents. This involves stop-word removal, lemmatization of the words converting them to their base forms, expansion of contractions, and removal of punctuations. The same pre-processing approach is applied to the query to get the pre-processed query.

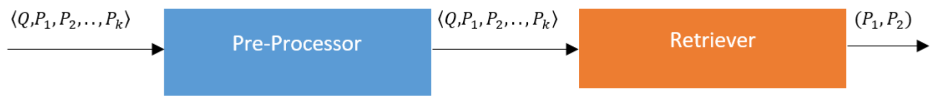

$\langle Q, P_1, P_2, .., P_k \rangle$
Pre-Processor
$\langle Q, P_1, P_2, .., P_k \rangle$
Retriever
(P_1, P_2)

Figure 2: Architectural diagram of the Paragraph Retreiver Module. The pre-processor will take an input(question and list of paragraphs) and output preprocessed paragraphs and question. The retriever will identify the top two paragraphs containing the answer to the question.

3.1.2 Retriever

a. BM25

The pre-processed user query and paragraphs are then passed to the retriever module which computes the paragraph scores represented by the similarity metric between the query and each of the paragraphs. The paragraph scores have been computed using standard state-of-the-art models like TF-IDF where the rarer words in the documents are given higher weightage. The similarity measure is computed as follows:

$$sim(q,p) = \sum_{t \in q} tf_{t,p} * idf_t \tag{1}$$

Where $tf_{t,p}$ is the frequency of term t in paragraph P and idf_t is the inverse document frequency of the term t.

$$idf_t = log(\frac{1+N}{1+df_t}) + 1 \tag{2}$$

Where N is the total number of documents in the corpus and df_t is the number of documents with term t.
Another model that we have tried is the Okapi BM-25 which incorporates the impact of document length along with the impact of the importance of words with TF-IDF. It has two hyper-parameters k and b controlling the impact of term frequency and document lengths respectively.

$$\sum_{t \in Q} idf_t * \frac{tf_{t,p}(k+1)}{k(1-b+b\frac{pl}{pl_{avg}}) + tf_{t,p}} \tag{3}$$

Where pl is the length of the paragraph and pl_{avg} represents the average document length.
The paragraphs are then ranked according to the paragraph scores and the paragraph with the maximum score is retrieved as the result which is then fed forward to the pipeline.

b. Retrieve and Re-rank Pipeline

The next experimented model was a retrieve and re-rank pipeline. This model uses two components – the Bi-encoder and the Cross Encoder – to enable passage retrieval and re-ranking.

The bi-encoder is a deep learning model that combines the encoder-decoder paradigm with the bi-directional Transformer architecture. It is used to retrieve relevant passages from a corpus of documents and works by encoding a query into a vector, which is then compared to the encoded documents. It retrieves the most relevant documents by computing the similarity score between the query vector and the document vectors. The cross-encoder is then used to re-rank the passages retrieved by the bi-encoder. It works by encoding the query and the retrieved passages into two separate vectors. These vectors are then compared to each other to determine the relevance of the retrieved passages. The Cross Encoder is able to capture the semantic relationships between the query and the passages, thus allowing for more accurate re-ranking.

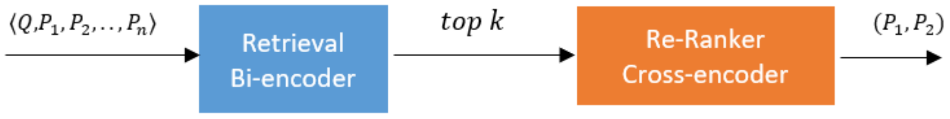


Figure 3: Architectural diagram of retrieve and re-rank pipeline

Cross-Encoders have a larger performance benefit because they pay attention to both the query and the document. However, it would take a while to score thousands of (query, document)-pairs. Hence, we use the retriever to compile a list of potential candidates, which the Cross-Encoder then reranks for more efficiency. Compared to the earlier retrieval techniques, this retrieval pipeline provided us with greater accuracy.

3.2 Question Answering

In this module, we have to answer the question from the paragraphs retrieved by the paragraph retriever. The module uses a span-based machine reading comprehension model which takes paragraph and question pair $< P, Q >$ at a time and predicts the start index and end index probabilities and answerability probabilities. The answerability verifier calculates the answerability scores from the predicted probabilities for both question paragraph pairs. After applying the answerability threshold, gets the answer span from the paragraph corresponding to the best answerability score.

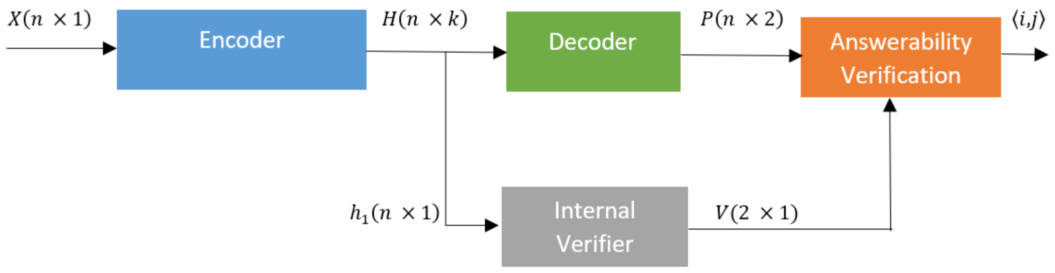


Figure 4: Architectural diagram of the question answering module. The encoder will take an input(paragraph + question) of size $n \times 1$ and output contextual embedding of size $n \times m$. Where n is the number of tokens in the input. The decoder will predict the start and end probabilities of size $n \times 2$. Internal verifiers take the pooled representation of the first token h_1 and predict the probability of answerability of the question. Answerability verification employs a threshold-based answerable verification and gives the answer span $< i, j >$

3.2.1 Encoder

Pre-trained Language Models(PrLM) like BERT, ALBERT, or ELECTRA is used as an encoder. The passage and question are first concatenated and tokenized (used pre-trained tokenizers corresponding to the pre-trained language model used) to generate input embedding $X = \{x_1, x_2, \ldots, x_n\}$. Input embedding will be the sum of its token embedding, position embedding, and token-type embedding. The input embeddings are then fed into the encoder to generate contextual representation $H = \{h_1, h_2, \ldots h_n\}$ with size $n \times m$.

Here h_i is the representation of i^{th} token having a size equal to the number of hidden units(m) in the last layer of the pre-trained language model.

3.2.2 Decoder

Used a fully connected layer with two output units and SoftMax activation as the decoder. The decoder will take the contextual representation of the input paragraph and question H as the input and predict the start and end probabilities s and e. The overall output will be of a size of $n \times 2$.

$$[s, e] = SoftMax(FFN(H)) \tag{4}$$

For training the span prediction we used cross entropy as the loss function as shown in equation 5. Where y_i^s and y_i^e are respectively ground-truth start and end positions of example i. P_a^s and P_b^s are the predicted start and end logits respectively($a \in y_i^s$ and $b \in y_i^e$). N is the number of examples.

$$L^{span} = -\frac{1}{N}\sum_{i=1}^{N}(log(P^s_{y_i^s} + log(P^e_{y_i^e}) \tag{5}$$

3.2.3 Internal Verifier

The internal verifier uses a fully connected layer with two output units and the SoftMax activation function which acts like a classifier. The pooled first token ([CLS]) representation $h_1 \in H$, as the overall representation of the sequence fed into the internal verifier to predict answerable and unanswerable probabilities. Cross entropy is used as the training objective of this module.

$$\hat{y_{i,k}} = SoftMax(FFN(h_1)) \tag{6}$$

$$L^{ans} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{2} y_{i,k} log(\hat{y_{i,k}}) \tag{7}$$

Where $k \in \{1, 2\}$, $y_{i,k}$ and $\hat{y_{i,k}}$ are the true and predicted probabilities. N is the number of samples used for the training.

3.2.4 Answerability Verification

This module employs a threshold-based answerability verification for checking whether the question is answerable or not. The module calculates answerability scores (S_{null}) from the output logits of the decoder and internal verifier. The answerability score is the linear combination of the scores calculated from the span logits from the decoder (S_{diff}) and the answerability logits from the internal verifier (S_{int})).

$$S_{null} = \beta_1 \times S_{diff} + \beta_2 \times S_{int} \tag{8}$$

The S_{diff} is the difference between the no-answer score (S_n) and has-answer score (S_y) calculated from the start index logits (s) and end index logits (e) as shown below.

$$S_y = max(s_i + e_j), \qquad 1 < i \leq j \leq n \tag{9}$$

$$S_n = s_1 + e_1 \tag{10}$$

$$S_{diff} = S_n - S_y \tag{11}$$

The S_{int} will be the difference between the no-answer logit ($logit_{na}$) and has-answer logit ($logit_{has}$) from the internal verifier.

$$S_{int} = logits_{na} - logits_{has} \tag{12}$$

After calculating the answerability score for both paragraphs take the minimum answerability score (minimum is best) and check whether it is less than an answerability threshold (δ). If it is, the model will predict the answer span from the corresponding paragraph and null string otherwise. The value of (δ) is determined with the help of the development set.

3.2.5 Training Methodology

The encoder is finetuned along with the decoder and internal verifier on the overall loss function as the combination of both span loss (L^{span}) and verification loss (L^{ans}) as shown in equation 13. The value of α_1 and α_2 are set as 1 and 0.5 respectively.

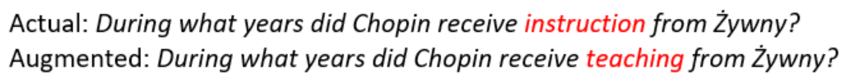
$$L = \alpha_1 L^{span} + \alpha_2 L^{ans} \tag{13}$$

3.2.6 Data Preprocessing

In the preprocessing part, removed Unicode characters from the paragraphs and questions. Found the end index from the answer text and start index and added a new column to the dataframe. For nonanswerable questions used the start index and end index as 0 (first token).

3.2.7 Data Augmentation

For increasing the dataset size we generated more question paragraph pairs by augmenting the questions using the WordNetAugmenter from textattack library. WordNetAugmenter uses WordNet for expanding the context of the word in the text. It works by retrieving the synonyms of words in the text and replacing them with words that are semantically similar.



Actual: *During what years did Chopin receive instruction from Żywny?*
Augmented: *During what years did Chopin receive teaching from Żywny?*

Figure 5: Example of actual question and augmented question using WordNetAugmenter. The augmenter replaced instruction with teaching.

4 Experiments and Results

4.1 Paragraph Retrieval

4.1.1 Retrieval Method Selection

We initially compared the results of BM25 retrieval and the Retrieval and re-rank pipeline. The gold passage's retrieval accuracy in the top k passages through the model is shown in the "Accuracy@top k" column.

Model	Accuracy@top1	Accuracy@top2	Accuracy@top3
BM25	64.1	73.2	75.1
Retrieve & Rerank	70.8	76.6	78.3

The retrieve and re-rank pipeline gave better results, hence we decided to move on with the method.

4.1.2 Selection of parameters

It was important to choose the number of candidate passages the bi-encoder would obtain so that there would be just the right amount for the cross-encoder to re-rank and not too many to negatively impact performance. We found a good number of candidate passages retrieved by the bi-encoder to be set as 10 and the maximum sequence length to be set as 512.

4.1.3 Model Selection

We then experimented with different pre-trained bi-encoder and cross-encoder models. The accuracies received are tabulated below.

Bi-Encoder	Cross-Encoder	Accuracy@top1	Accuracy@top2	Accuracy@top3
multi-qa-MiniLM-L6-cos-v1	ms-marco-MiniLM-L-6-v2	70.8	76.6	78.3
all-MiniLM-L6-v2	ms-marco-MiniLM-L-6-v2	73.3	77.6	83.1
all-MiniLM-L6-v2	qnli-distilroberta-base	68.3	79.2	80.4
multi-qa-mpnet-base-dot-v1	ms-marco-MiniLM-L-6-v2	**78.0**	**83.2**	**84.0**

4.2 Question Answering

4.2.1 Model Selection

We tried pre-trained language models $BERT_{base}$, $ALBERT_{base}$, $ELECTRA_{base}$ along with their pre-trained weights as the encoder. The encoder is then finetuned along with the decoder and internal verifier(IV) with all the official hyperparameters of the pre-trained language model. We compared the results with the model trained without the internal verifier(IV). The validation F1 scores obtained for training with different pre-trained language models with and without internal verifier are reported in table 1 along with the training time and average inference time. $ELECTRA$ showed the best F score among all three models. We used the ADAM optimizer for all training with an initial learning rate of 2e-5 and an L2 weight decay of 0.01. The batch size is used as 8 and the maximum number of epochs is set as 5 with patience of 2 in all the experiments. In all the experiments we used a maximum sequence length of 512 and validation data from themes not used for the training.

Model	F1 Score	Training Time	Avg Prediction Time
BERT	77.2	10hr 20min	140.15 ms
BERT+IV	79.1	10hr 50min	155.77 ms
ALBERT	78.9	10hr 05min	**121.43 ms**
ALBERT+IV	82.1	10hr 35min	135.9 ms
ELECTRA	82.6	10hr 10min	125.7 ms
ELECTRA+IV	**84.3**	10hr 40min	141.2 ms

Table 1: F1 score obtained on all the experiments

All the above-mentioned experiments are run in the basic google colab environment with an available RAM of 12.68GB and disk space of 78.19GB. The resource usage for all the experiments is given in table 2. Due to resource and training time constraints, we couldn't go for the large language models.

4.2.2 Selection of Max Length of Input Sequence

The results from the Model selection showed the best results with ELECTRA+IV. So we decided to go with the ELECTRA+IV model. During the data exploration of the training dataset provided to us, we found out that the input sequence length greater than 350 tokens is less than 1%. So to improve the efficiency of the prediction we reduced the length of the model's input sequence length from 512 to 384. The average prediction time improved from 141.2 ms to 79.38 ms with a negligible reduction in answer prediction F1 scores.

Model	RAM(GB)	Disk(GB)
BERT	4.13	23.59
BERT+IV	4.65	24.12
ALBERT	4.24	23.23
ALBERT+IV	4.53	23.59
ELECTRA	4.26	24.68
ELECTRA+IV	4.89	24.81

Table 2: RAM and disk usage for training

4.2.3 Finding Non-Answerability Threshold

In section 3.2.4 we explained the non-answerability verification methodology from the answer span logits from the decoder and has-answer logits from the internal verifier. If the answerability score (S_{null}) is less than a threshold δ, the question is not answerable. For finding the δ we tried different beta values and searched for the best answerability score as the δ which maximizes the F1 score of answerable prediction on the validation data. From the different random searching, we found the best combination of values as β_1 0.1, β_2 1.2, and δ -0.74.

4.2.4 External Verifier

Apart from the internal verifier we trained a separate answerability verifier with the encoder as pre-trained language models mentioned above and the verification decoder as a fully connected layer with two output units and the SoftMax activation function. The input sequence after encoding by the pre-trained language models the pooled first token ([CLS]) representation $h_1 \in H$, passed into the decoder for answerable and non-answerable probabilities. The model is trained with cross-entropy loss similar to the internal verifier. The answerability score for the external verifier (S_{ext}) is calculated similarly to S_{int}.

In the answerability threshold estimation, we tried the different combinations S_{diff}, S_{ext}, and S_{int}. We found that there is no extra gain in the answerability prediction F1 scores while adding the external verifier and we discarded it for avoiding extra complexity.

4.2.5 Fine Tuning on New Dataset

For finetuning with the newly received dataset we used the data augmentation method mentioned in section 3.2.6. We were able to double the size of the dataset. After finetuning the ELECTRA model on the augmented dataset for 3 epochs we were able to receive a training F1 score of 72.6% and a validation F1 score of 56.4%.

5 Conclusion

The results of our experiments indicate that our system is capable of achieving high performance in terms of both paragraph retrieval and question answering. Overall, we believe that our work on the project has made significant progress towards the goal of building a robust Question Answering system with the given constraints. We mainly focused on pre-trained language models as they are better at learning contextual representations.

In conclusion, the model created using ELECTRA+IV and cross-encoder performed better in terms of accuracy without causing much difference in terms of efficiency when compared to its counterparts of BERT and ALBERT using cross-encoder/ bi-encoder. The models did not cause much difference in accuracy when the max sequence length was reduced from 512 to 384 while it showed significant improvement in performance. Answerability verification methods were a success in improving the accuracy of both tasks.

References

[1] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, “Squad: 100,000+ questions for machine comprehension of text,” 2016. [Online]. Available: https://arxiv.org/abs/1606.05250

[2] P. Rajpurkar, R. Jia, and P. Liang, “Know what you don’t know: Unanswerable questions for squad,” 2018. [Online]. Available: https://arxiv.org/abs/1806.03822

[3] M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer, “Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension,” 2017. [Online]. Available: https://arxiv.org/abs/1705.03551

[4] A. Trischler, T. Wang, X. Yuan, J. Harris, A. Sordoni, P. Bachman, and K. Suleman, “Newsqa: A machine comprehension dataset,” 2016. [Online]. Available: https://arxiv.org/abs/1611.09830

[5] K. M. Hermann, T. Kočiský, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, “Teaching machines to read and comprehend,” 2015. [Online]. Available: https://arxiv.org/abs/1506.03340

[6] R. Kadlec, M. Schmid, O. Bajgar, and J. Kleindienst, “Text understanding with the attention sum reader network,” 2016. [Online]. Available: https://arxiv.org/abs/1603.01547

[7] W. Wang, N. Yang, F. Wei, B. Chang, and M. Zhou, “Gated self-matching networks for reading comprehension and question answering,” in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 189–198. [Online]. Available: https://aclanthology.org/P17-1018

[8] B. Dhingra, H. Liu, Z. Yang, W. W. Cohen, and R. Salakhutdinov, “Gated-attention readers for text comprehension,” 2016. [Online]. Available: https://arxiv.org/abs/1606.01549

[9] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, “Bidirectional attention flow for machine comprehension,” 2016. [Online]. Available: https://arxiv.org/abs/1611.01603

[10] Y. Cui, Z. Chen, S. Wei, S. Wang, T. Liu, and G. Hu, “Attention-over-attention neural networks for reading comprehension,” in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2017. [Online]. Available: https://doi.org/10.18653%2Fv1%2Fp17-1055

[11] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, “Deep contextualized word representations,” 2018. [Online]. Available: https://arxiv.org/abs/1802.05365

[12] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving language understanding by generative pre-training,” 2018.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” 2018. [Online]. Available: https://arxiv.org/abs/1810.04805

[14] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, “Xlnet: Generalized autoregressive pretraining for language understanding,” 2019. [Online]. Available: https://arxiv.org/abs/1906.08237

[15] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “Roberta: A robustly optimized BERT pretraining approach,” *CoRR*, vol. abs/1907.11692, 2019. [Online]. Available: http://arxiv.org/abs/1907.11692

[16] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, “ALBERT: A lite BERT for self-supervised learning of language representations,” *CoRR*, vol. abs/1909.11942, 2019. [Online]. Available: http://arxiv.org/abs/1909.11942

[17] K. Clark, M. Luong, Q. V. Le, and C. D. Manning, "ELECTRA: pre-training text encoders as discriminators rather than generators," *CoRR*, vol. abs/2003.10555, 2020. [Online]. Available: https://arxiv.org/abs/2003.10555

[18] X. Liu, W. Li, Y. Fang, A. Kim, K. Duh, and J. Gao, "Stochastic answer networks for squad 2.0," 2018. [Online]. Available: https://arxiv.org/abs/1809.09194

[19] M. Hu, F. Wei, Y. Peng, Z. Huang, N. Yang, and D. Li, "Read + verify: Machine reading comprehension with unanswerable questions," 2018. [Online]. Available: https://arxiv.org/abs/1808.05759

[20] S. Back, S. C. Chinthakindi, A. Kedia, H. Lee, and J. Choo, "Neurquri: Neural question requirement inspector for answerability prediction in machine reading comprehension," in *International Conference on Learning Representations*, 2020.

[21] Z. Zhang, J. Yang, and H. Zhao, "Retrospective reader for machine reading comprehension," 2020. [Online]. Available: https://arxiv.org/abs/2001.09694

[22] A. Ahmad, N. Constant, Y. Yang, and D. Cer, "ReQA: An evaluation for end-to-end answer retrieval models," in *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*. Association for Computational Linguistics, 2019. [Online]. Available: https://doi.org/10.18653%2Fv1%2Fd19-5819

[23] C. Clark and M. Gardner, "Simple and effective multi-paragraph reading comprehension," 2017. [Online]. Available: https://arxiv.org/abs/1710.10723

[24] M. M. Hoque and P. Quaresma, "An effective approach for relevant paragraph retrieval in Question Answering systems," https://ieeexplore.ieee.org/document/7488040, 2015.

[25] C. Macdonald and N. Tonellotto, "On Approximate Nearest Neighbour Selection for Multi-Stage Dense Retrieva," https://arxiv.org/pdf/2108.11480.pdf, 2021.

[26] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," 2019. [Online]. Available: https://arxiv.org/abs/1908.10084