Extractive Question Answering On Large Structural Engineering Documents

Stanford CS224N Custom Project

Adam Banga
Department of Civil Engineering
abanga19@stanford.edu

Thomas Sounack
Department of Mechanical Engineering
tsounack@stanford.edu

Abstract

Our project aims to address the challenge of efficiently searching through lengthy construction documents for specific information, in which using typical generative models is not desirable due to potential hallucinations. As a result a two part extractive question answering model will be implemented. The first part will be comprised of an information retrieval(IR) model using methods such as the BM25 algorithm, cosine similarity between query and documents, and BM25 with MuGI (multi-text generation integration). For the second portion, we are using an encoder-only architecture as the extractive question answering (EQA) model, considering models like BERT, RoBERTa and Longformer for optimal performance. Preliminary testing on a small sample of engineering documents presents a baseline end to end accuracy of .18 in which BM25 with k=1 and a Bert Model was used. Upon further testing with BM25 with MuGI and a RoBERTa model an end to end accuracy of .2 was reached. Ultimately BM25 with Mugi no reranking presented moderately better results than baseline BM25 and in reference to the extractive question answering portion the RoBERTa model performed best.

1 Key Information to include

- Mentor: Soumya Chatterjee
- External Collaborators (if you have any): NA
- Sharing project: No
- Individual Contributions:
  - Adam: I focused predominantly on the information retrieval portion of the pipeline namely implementing, BM25, cosine similarity, and BM25 with MuGI. As a result portions of the report pertaining to information retrieval of paragraphs are predominantly written by myself. All other aspects of the project were shared and split evenly.
  - Thomas: I focused predominantly on the extractive question answering portion of the pipeline implementing BERT, RoBERTa, and Longformer. As a result portions of the report pertaining to extractive question answering are predominantly written by myself. All other aspects of the project were shared and split evenly.

2 Introduction

The construction industry is a document reliant industry. As a result there are often thousands of different text based documents each varying from a few pages to hundreds of pages. As a result finding information within these documents can be of great difficulty. Typically the "command f" function on pdf viewers is highly utilized. However, this function doesn't allow for a semantic search but rather only does a basic string comparison.

The primary challenge in this endeavor lies in the sheer volume of documents to be searched, often spanning hundreds of pages. For instance, the bespoke test dataset crafted for assessing the pipeline outlined in this paper encompasses 300 pages. Consequently, many Language Model (LLM) architectures struggle due to their limited context window, failing to encompass an entire document for processing. Recently, breakthroughs from companies like Google, Anthropic, and OpenAI have introduced LLMs with significantly expanded context windows capable of handling extensive documents. However, such models may be cost-prohibitive and inaccessible. Moreover, many architectures tackling this task, such as Retrieval Augmented Generation (RAG) models, are susceptible to hallucinations, posing risks unacceptable for industrial applications. Many professionals in this domain are retrieving information within safety-critical environments, such as construction sites, where incorrect information could lead to potentially fatal outcomes during structure assembly. As a result, the motivation behind this project is to develop a model capable of processing natural language queries and pinpointing exact textual locations where answers reside, circumventing the pitfalls associated with generative models.

In this pursuit, our model has been broken down into two segments: an information retrieval portion and an extractive question answering model. For the information retrieval portion three different methods will be explored including the BM25 algorithm(Robertson et al., 1994), cosine similarity between embeddings of query and documents, and finally BM25 with MuGI (multi-text generation integration)(Zhang et al., 2024). The most promising method based on current research presents BM25 with MuGI, which involves generating pseudo queries to provide the first query with more context with the aims for BM25 to be more successful. Each information retrieval method will be designed to output k-relevant paragraphs to be passed to the extractive question answering model.

For the extractive model, an encoder-only architecture is preferred as we do not wish to generate an output. We implement BERT, ROBERTA and Longformer to return a span for each of the k-relevant paragraphs.

3 Related Work

At the time of proposal, two distinct methodologies for achieving the task of extractive question answering are prominent within contemporary research. The first method involves looking at attention mechanisms within a model to understand the tokens that bear greater weight in generating a response to a query. Generative AI models exhibit proficiency in text generation, yet they are susceptible to hallucinations as mentioned previously. Therefore, a deeper comprehension of the corpus-influencing factors behind a model's output is necessary. The investigation of attention weights and their interpretability within the model has been extensively explored, as evidenced by studies such as Jain et al.'s "Attention is not not Explanation" (Jain and Wallace, 2019) and Wiegreffe et al.'s "Attention is not not Explanation" (Wiegreffe and Pinter, 2019). These works delve into the interpretability of attention, exploring whether attention weights can elucidate the rationale behind a model's text generation process, thereby representing a significant avenue in addressing this challenge.

A second methodology involves the deployment of extractive question-answering models, which receive an input context and a question and output the beginning and end positions of the answer within the provided context. Models like BERT(Devlin et al., 2019) have demonstrated good performance for this task. There has also been research namely by Xu et al combining both of the aforementioned methods of attention with an extractive question answering model (Xu et al., 2021). This approach by itself is not sufficient for us as these methods are limited by the models' context lengths. Context lengths of BERT models tend to be 512 tokens, which is very little compared to the 300 page-long documents we wish to process. Simply designing an architecture with a larger context length would be intractable at this scale, since attention scales quadratically with sequence length.

4 Approach

To tackle the challenge of extractive question answering on lengthy documents, the problem was divided into two parts: information retrieval and extractive question answering. Due to length of engineering documents, conventional models such as BERT lack a sufficiently large context window (512 tokens) to process the entire document alongside a query. Therefore, an approach was adopted to segment the document into paragraphs of less than 512 tokens, employing IR to identify the k most relevant paragraphs, and subsequently utilizing an EQA model, with a strategy to choose the right answer among the k outputs.

Information Retrieval (IR): Three methods were explored for the IR stage. The process involves breaking the long document into paragraphs and using an algorithm to find the k-most relevant paragraphs for a given query.

The first method which was utilized as a baseline was the BM25 algorithm (Robertson et al., 1994). BM25 is a ranking function used in IR to score and rank documents based on their relevance to a given query, utilizing term frequency and document length normalization. For this methodology strings are tokenized into words based on white spaces and then these are passed to the BM25 algorithm as well as tokenized documents.

In the second approach the query is embedded as well as each paragraph of the document. Cosine similarity is then computed between the query and each paragraph embedding, and the k-most relevant paragraphs are returned. The embedding approach aims to capture semantic meanings which the BM25 algorithm cannot leverage due to its parametric nature. For this approach a BERT model named:'bert-large-cased-whole-word-masking' was used to tokenize and create embeddings. The query was also embedded using the same model and cosine similarity was computed on these resulting embeddings.

The final method employed in this study integrates BM25 with MuGI (Zhang et al., 2024). This approach entails enriching the initial query with passages generated by a Large Language Model (LLM), aiming to expand the query space and facilitate the identification of relevant paragraphs within the document during the BM25 search process. To generate pseudo queries, we utilized the GPT-3.5-turbo-1106 model, prompting it with the instruction: "Generate a new query relevant to the following: 'original$_{query}'.Ensurethequeryisconcise, informative, andclear, resemblingtheoriginalquery.Notethatthequeryaim$
$tunedtooptimizemodelperformance.Thesesupplementaryqueriesarethencombinedwiththeinitialquerytoformafin$
$evaluatesparagraphrelevancebasedoncosinesimilarityembeddingswiththedocumentsandtheaugmentedquery, mirr$

Extractive Question Answering (EQA): EQA consists of outputting a span given a query and a context. This translates into using an encoder model to infer the probability of each of the context's tokens being the start/end token. For both outputs, we then take the argmax to determine the predicted start and end indexes, which we can use to index into the context to return the answer. The model can also return the token [CLS], meaning that the provided context is not sufficient to provide an answer. For this component, we will use pre-trained models from the HuggingFace library. We will be comparing three encoders: BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019) and Longformer (Beltagy et al., 2020). BERT and RoBERTa typically employ a maximum sequence length of 512 tokens, with computational requirements increasing quadratically with this value. Conversely, Longformer scales linearly with maximum sequence length. Exploring larger sequence lengths may improve the success rate of the IR algorithm by reducing the number of paragraphs to choose from (with more tokens per paragraph available, the documents is cut into less paragraphs).

Linking IR and EQA Model: After labeling k relevant paragraphs with the IR algorithm, each is passed into our extractive QA model to generate predictions. Ideally, we aim for the model to output [CLS] for all but the paragraph containing the answer, but consistent results may not be achieved. To select the correct prediction from the k outputs, one option is to use the highest-ranked non-[CLS] paragraph from IR. Alternatively, a new model can be prompted with the query and prediction to determine if it answers the query.

At the time of this report, linking has not been extensively researched and represents an area for future exploration within this pipeline. Ultimately, during end-to-end testing, the top paragraph ($k = 1$) retrieved from the information retrieval segment of the pipeline was passed to the extractive question answering module.

Baselines: For our baseline, we use BM25 for information retrieval and BERT as the extractive QA model. Using k=1 for the BM25 algorithm, we extract the paragraph most relevant to each query, and then pass this paragraph along with the query through BERT to get an answer. The results for our baseline can be found in the experiments portion of the report.

5 Experiments

5.1 Data

The baseline dataset originates from a 300-page Lakers Practice Facility construction contract. Presently, the dataset encompasses 50 samples for testing, a notable increase from the initial 21 samples during the baseline testing phase, with ongoing plans for further expansion. Designed to assess both the information retrieval and extractive question answering components of the pipeline independently and in tandem, this dataset is structured to include a query, the corresponding paragraph containing the queried information excerpted from the 300-page document, and a precise output indicating the answer's location within the paragraph. Such design enables comprehensive testing of the information retrieval component using the query, the entire PDF document, and the designated paragraph as the gold standard. Likewise, the extractive question answering component can be evaluated using the query, the gold-standard paragraph, and the extracted answer string. Moreover, the pipeline's end-to-end performance can be rigorously assessed using this dataset.

5.2 Evaluation method

We evaluate three score: the accuracy of the information retrieval methods (ie. when the right paragraph is used), the partial match of the extractive QA model (since we just want to show the right emplacement to the user), and the accuracy of the whole implementation end-to-end (ie. the right paragraph is identified by BM25 and the right answer is returned).

To assess information retrieval accuracy, we employ an iterative approach using an accuracy-based metric. We evaluate the top k most similar paragraphs against a designated "gold" paragraph. Accuracy is determined by whether the gold paragraph is found within the first most similar paragraph, then extends to the top two, and progressively up to the top k paragraphs. If the gold paragraph is found within the top n out of k paragraphs, it scores 1; otherwise, it scores 0. Aggregated scores across all data points are then normalized to produce a value between 0 and 1 at each respective index, indicating overall accuracy.

For the extractive QA model, we evaluate the partial match by looking at the ground truth start and end token indexes, as well as the outputted start and end token indexes. If there is an intersection between the two intervals, we return 1, otherwise we return 0. The partial match score is then computed by averaging these results. Note that we test on all our data samples using the correct paragraph, regardless of whether BM25 identified the right paragraph.

5.3 Experimental Details

For BM25, we tokenize by isolating each word denoted by white space and use k=1. For the extractive QA model, we use BERT pretrained on the squad dataset (Rajpurkar et al., 2018), with truncation and max length padding for max length = 512 tokens.

5.4 Results

Table 1 below presents the baseline results obtained during the project milestone. These results encompass the utilization of BM25 in its default configuration, a BERT-based extractive Question Answering model, and a straightforward linking technique involving the top paragraph retrieved by BM25 passed to BERT. Notably, our baseline accuracy for the BM25 algorithm stands at 0.43. However, upon closer examination in Table 2, the accuracy of BM25 for $k = 1$ is recorded at 0.38, which falls below our milestone baseline. This discrepancy can be attributed to the expansion of our test dataset from 21 samples during the milestone to the current 50 samples curated at the time of the final project assessment.

BM25	BERT	End-to-end Model
Accuracy: 0.429	Partial Match: 0.476	Accuracy: 0.238

Table 1: Experimental results for the baseline

For the information retrieval portion of the pipeline, Bm25, cosine similarity, and various permuations of BM25 with MuGI were tested. For BM25 with Mugi, we tested with generation of 3 additional passages from the GPT model and with 5 passages from GPT. We also test the final reranking portion of the pipeline and toggled that as another hyper parameter. Below is a table with our results based on our evaluation metric discussed before. Note "Mugi 5Q NR" means BM25 with Mugi with 5 additional GPT queries and with no reranking.

	BM25	Cos-Similarity	MuGI 3Q	MuGI 3Q NR	MuGI 5Q NR
Accuracy in K=1	.38	.14	.18	.44	.48
Accuracy in K=2	.50	.18	.24	.56	.56
Accuracy in K=3	.54	.20	.30	.60	.56

Table 2: Accuracy for different information retrieval methods for k=3

It is intriguing to observe that our baseline BM25 algorithm demonstrated remarkably strong performance compared to other methods. Conversely, cosine similarity, which relies on the similarity of embeddings between queries and paragraphs, performed the least effectively. This observation prompted us to explore an alternative approach: employing BM25 with MuGI without reranking, as reranking essentially involves reordering based on cosine similarity. Given the suboptimal performance of cosine similarity, experimenting with MuGI without reranking presented an intriguing possibility, resulting in a significant improvement over traditional MuGI implementations. Furthermore, incorporating 5 queries from GPT rather than 3 queries did not notably impact the accuracy of the MuGI retrieval.

The embeddings derived from the BERT model ('bert-large-cased-whole-word-masking') may not yield optimal representations for the paragraphs within the test document. Utilizing a more specialized engineering embedding model could potentially enhance the results of the cosine similarity approach. Given that the BERT embeddings are likely trained on dissimilar types of text, applying them to a distinct construction/engineering text corpus may result in subpar embeddings. It is anticipated that the embedding space for each paragraph will exhibit proximity, potentially leading to degraded performance of cosine similarity.

The information retrieval component of the pipeline demonstrated a maximum performance of approximately 50% for $k = 1$. While achieving a 50% chance of retrieving the correct page out of 300 pages is noteworthy, there remains significant room for improvement before adoption in industry becomes viable. Enhancing performance entails fine-tuning our prompt and selecting a more robust embedding model, which appear to offer the greatest potential for improvement. An intriguing avenue for future research could involve fine-tuning embeddings specifically on engineering documents and investigating resultant performance changes, thus representing a promising direction for further exploration.

For the extractive question answering part, we use three decoders: BERT, RoBERTa and Longformer. We look at their implementation independantly of BM25, by feeding the correct context and the query to the model. Their results are summarized in the table below:

EQA comparison	BERT	RoBERTa	Longformer
Partial Match	0.48	0.52	0.2

Table 3: Partial match score for EQA models

We can see that using RoBERTa provides a slight increase over the BERT model. Longformer, however, does not perform as well. Interestingly, its outputs are very short compared to its two counterparts, and often miss the context by a few tokens only.

Now that both components of our pipeline are implemented, we can test the performance of our end-to-end pipeline with different configurations. For the linking portion of our algorithm, we choose the first page returned by the information retrieval algorithm and pass it to the extractive question answering model to generate the answer. The results are summarized in the following table:

We can see that the best performing configuration is MuGI 5Q NR with RoBERTa. As anticipated, RoBERTa and MuGI 5Q NR are the best performing components on their own and form the highest scoring pipeline.

Accuracy	BM25	Cos-Similarity	MuGI 3Q	MuGI 3Q NR	MuGI 5Q NR
BERT	.18	.07	.09	.18	.18
RoBERTa	.16	.04	.06	**.2**	**.2**
Longformer	.08	.03	.04	.08	.1

Table 4: End-to-end pipeline accuracy

These results are lower than expected, we would have hoped to observe better accuracies for the end to end pipelines. We will discuss possible reasons for this performance in the following section.

6 Analysis

Overall, we have seen that the performance obtained with our implementation does not reach the industry standards. This is most likely due to the length of the document, but also to its technical nature, as mentioned in the previous part.

We can perform error interrogation to understand the failure modes of our implementation.

For BM25, the algorithm tends to perform worse when the objective page is sparse. For instance, a query from the dataset is "What type of contract is the Lakers Practice Facility under?". BM25 returns the first page as its first prediction, although the right page is the third page. In this example, the third page is sparse, and contains little information. The first page, however, introduces the project and is very dense. Page 50, on the other hand, is very dense and contains several mentions of the word contractor. All the queries linked to this page are correctly traced back by BM25, as they are queries regarding the contractor's responsibilities.

For the extractive QA component, we can also identify failure patterns. Similarly to what was observed before, sparse pages tend to show better results. In this case however, it is most likely a matter of probabilities: there is a higher chance of extracting the correct span when there is less text to choose from. On the other hand, it is interesting to note that the encoder offers a very good performance when asked to retrieve numbers (for instance with the query "What is the telephone number of the Architect?").

7 Conclusion

During this project, we implemented a pipeline for extractive question answering on long engineering documents by implementing and comparing different information retrieval methods, extractive question answering models, and how to link them.

We found that while we were able to achieve relatively good performance given the length of our document and the complexity of the task, the performance that we obtained still is not on par with industry standards. Improvements have to be made before this can be implemented in real use.

There are several areas of improvement that might lead to an improved performance. We mentioned that generating embeddings for such technical documents might not be effective if the data that the embedding model trained on was general. Using an engineering-specific embedding model, or manually labelling data to fine-tune an embedding model might lead to an increase in performance. If this allows the BM25 algorithm to perform much better than it currently does, we could then split the documents in smaller paragraphs - which would make it easier for the extractive QA model to perform well.

References

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not explanation.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

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

Stephen Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at trec-3. pages 0–.

Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not explanation.

Peng Xu, Davis Liang, Zhiheng Huang, and Bing Xiang. 2021. Attention-guided generative models for extractive question answering.

Le Zhang, Qian Yang, and Yihong Wu. 2024. Mugi: Enhancing information retrieval through multi-text generation integration with large language models.

A Appendix (optional)

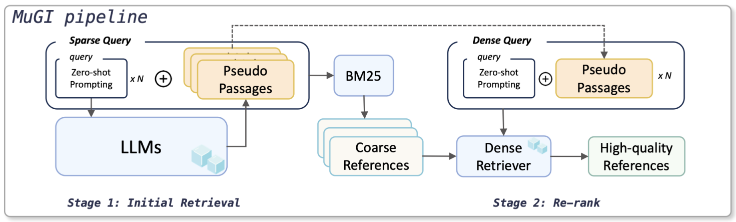

Figure 1: "Method overview of MuGI. Left part is initial retrieval using sparse retriever BM25, right part indicates re-ranking output from first stage given enhance Dense Query using a dense retriever."- Zhang et al. (2024)