Domain-Specific Extractive Question Answering from Material Science Literature

Anagha Savit
anagha.savit2001@gmail.com
Indian Institute of Technology Bombay
Thrissur, Kerala, India

Vidushi Verma
vidushimedhavi@gmail.com
Indian Institute of Technology Bombay
Gurgaon, Delhi, India

Aruja Khanna
arujakhanna12@gmail.com
Indian Institute of Technology Bombay
Mumbai, Maharashtra, India

ABSTRACT

Performing a thorough literature review is among the most cumbersome tasks when carrying out a research project. Students often invest significant amounts of time and effort in combing through scientific texts to uncover solutions to their inquiries, which can result in a waste of valuable resources. We aim to create a system that enables students to quickly and efficiently locate the most pertinent research paper and its corresponding section for their inquiries. Our pipeline takes as input research papers and queries from the user and, if possible, returns the answers and the most relevant research paper. We break down our task into two primary subtasks: paragraph retrieval and question answering. By using a bi-encoder and cross-encoder pipeline, we extract the most relevant passages from the input scientific texts that corresponds to the user's query for the paragraph retrieval subtask. Once we have retrieved the relevant paragraph, our question answering model provides the user with the answer to their query and the corresponding research paper, if the answer is present in the input papers. We fine-tune the pre-trained models on data obtained from various material science research papers and by performing webscraping to create our model that is tailored specifically for material science scientific text. The code for our final pipeline can be found at https://github.com/anaghasavit/material-science-extractive-QA.git

CCS CONCEPTS

• **Information systems → Question answering**.

KEYWORDS

NLP, Paragraph Retrieval, Question Answering, Transformers, Material Science, Sentence Transformers

ACM Reference Format:
Anagha Savit, Vidushi Verma, and Aruja Khanna. 2023. Domain-Specific Extractive Question Answering from Material Science Literature. In *Proceedings of ACM Conference (KDD-UC'23)*. ACM, New York, NY, USA, 6 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Carrying out literature review is an incredibly time-consuming and laborious task. It frequently results in the expenditure of significant resources in sifting through irrelevant texts in search of answers to queries. Information extraction is especially important in material science as it is a multidisciplinary field. Researchers need to collect information from a wide variety of sources, such as research papers, journals, and patents. Moreover, the field of material science consists of a diverse range of material categories and properties, resulting in highly heterogeneous and extensive data. Text data extraction methods can be used to automatically identify and extract key information from these sources, such as material properties, experimental methods, and results. This can save researchers substantial time so that they can focus on analyzing and interpreting the information, rather than having to search for it themselves.

Several efforts have been made to automate the process of scientific text data extraction in the material science domain. Researchers have made use of machine learning and NLP to extract information and develop predictive models for zeolite synthesis[1], adapted named entity recognition, relation extraction, and knowledge graphs to retrieve information from polymer, general material science literature[2, 3], and even fine-tuned the BERT architecture on a large corpus of materials science publications for extracting information from materials science text [4]. However as outlined in [5, 6], there is still a need for more developed and accurate domain-specific models for information extraction from material science publications.

The objective of our research is to bridge this gap through the development of a highly efficient and accurate pipeline for material science domain-specific question answering. We divide our task into two major subtasks: paragraph retrieval followed by question answering. Paragraph retrieval refers to the extraction of the most relevant paragraph for the user's query while question answering refers to the extraction of the answer from that particular paragraph. The final pipeline consists of a bi-encoder+cross-encoder for paragraph retrieval and a QA model for question answering.

This paper is organized as follows: part 2 outlines the overall pipeline; part 3 elaborates on the creation of a material science-specific QA dataset and the data augmentation methods used; parts 4 and 5 discuss the paragraph retrieval subtask, specifically the bi-encoder and cross-encoder evaluation and fine-tuning; part 6 details the question answering subtask, the QA model evaluation and fine-tuning; part 7 is the discussion of possible future work along with the conclusion.

2 PIPELINE

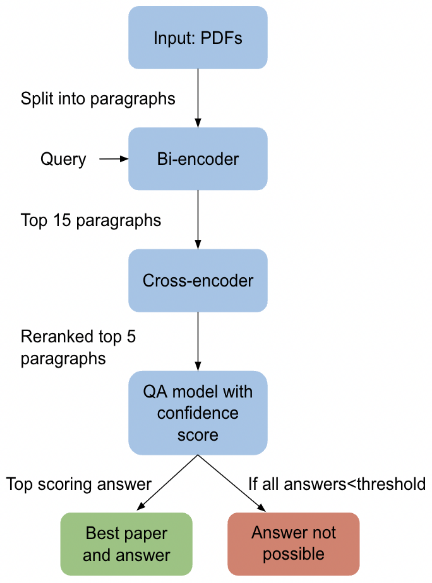
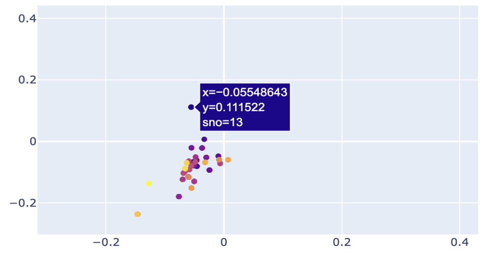


Figure 1: Flowchart of Overall pipeline

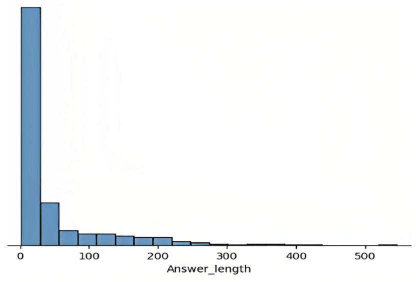
- The first step in the pipeline is to convert the input pdfs of research papers into txt files and split these txt files paragraph wise.
- The next step is to obtain the embeddings for all the paragraphs and the query using the bi-encoder. The bi-encoder then retrieves the top 15 passages for a given query, using semantic search. This reduced subset of passages is propagated to the cross-encoder.
- The top 15 passages obtained are re-ranked using a cross-encoder model which predicts a score between 0 and 1 for each passage. The top 5 passages with a relatively higher score among the top 15 passages are obtained.
- In the answer extraction step the QA model applied to each of these top 5 question-passage pairs, returns the answer from each passage along with the associated confidence score.
- The answer with the highest confidence score among the 5 passages is returned. If all the scores are below a pre-defined threshold, then "answer not possible" is returned.

3 DATA COLLECTION

Our dataset is created by a combination of manual and automated efforts. It consists of a large corpus of rows containing context, question, answer possible(binary values), answer text, answer start character, and answer end character. The contexts are chosen from highly esteemed material science research papers and journals and by scraping material science-specific Wikipedia pages. The next step is to create question-answer pairs for these contexts. Some of the question-answer pairs are created manually. Additionally, we also make use of an answer-aware question generation module in order to generate question-answer pairs. It consists of two separate modules: the first module identifies certain key phrases in the contexts, and the second module generates questions having these key phrases as the answers. We use a sequence-to-sequence question generator based on the pre-trained t5-base model [7] for setting up this question generation system.



(a)



(b)

Figure 2: (a) Paragraph wise clustering of the embeddings of the cluster centroids after projection to 2D space via PCA for a subset of the dataset (b) Distribution of answer lengths

3.1 Data Augmentation

We use three different strategies to augment our dataset. Synonym substitution is one of the methods used. First, the words in question answer pairs that are adjectives are identified. This is done by using the nltk (natural language tool kit) library in python. Once the adjectives are identified, they are replaced with synonyms obtained from the same library. The meaning of the text is maintained while adding variation to the dataset. With this new dataset, our model will be more robust, generalizable, and better able to handle deviations.

The second approach used is back translation. In this approach, the query is translated into an intermediate language, French in our case, and then translated back into English. This process of back translation often results in rephrasing of the query once it is translated back into the source language. In the case where the back translated query is a duplicate of the original, it is dropped. The rephrased queries are added to the dataset, helping to reduce overfitting and also lending it several of the same benefits as synonym substitution. We make use of the MarianMT model [8] for neural machine translation and back translation.

Finally, we identified that our dataset contains many more questions for which an answer is possible (TRUE questions) than it does questions for which an answer is impossible (FALSE questions). In order to balance out our dataset we carry out negative sampling using the results of the bi-encoder. The exact procedure is detailed in the next section.

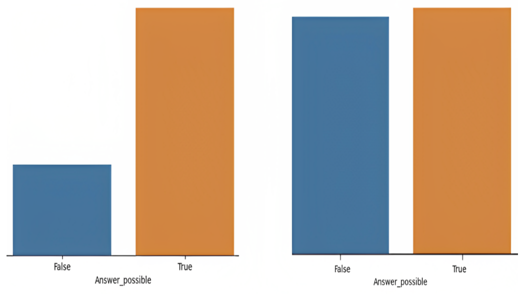


Figure 3: Distribution of TRUE and FALSE questions before and after negative sampling

4 BI-ENCODER

Ranking the passages based on literal matches between the query's words and the document's words, also called lexical search, is not robust to different spellings, acronyms, or synonyms. Hence, in our pipeline, the given passages and the query are converted into vector representations in semantic space. Semantic search (also known as dense retrieval) retrieves the passage embeddings near the query embeddings in this vector space. This measure of nearness can be cosine similarity, Euclidean distance, or dot product.

To obtain embeddings for both the passages and the query, we make use of a bi-encoder model that utilizes BERT [9] as an encoder. The bi-encoder model architecture pools the token embeddings generated by BERT to create new sentence embeddings. With the bi-encoder, we can retrieve the top 15 relevant passages for a query, based on the dot-product similarity measure. The working of the bi-encoder model is depicted in Figure 4.

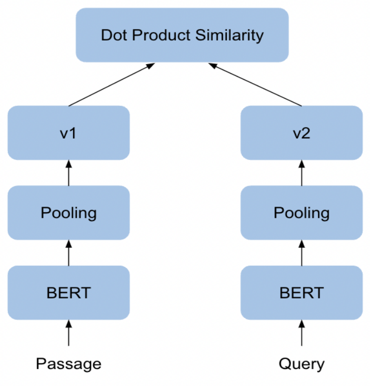


Figure 4: Bi-encoder working

In order to determine the best model, we evaluate the zero-shot performance of multiple different pre-trained bi-encoder models [10, 11, 12, 13] on our dataset. These pre-trained models are publicly available at https://www.sbert.net/.

The SBERT [14] models are evaluated on the basis of two criteria:

- Mean Reciprocal Rank (MRR): MRR corresponds to the reciprocal of the rank of the ground truth passage in the top K retrieved passages.
- Hits@K: Hits@K indicate whether the top K passages contain the ground truth passage or not.

The zero-shot performance is evaluated considering three different values of K (5/10/15). We also analyze the memory requirements of the models based on its parameters and buffers. The results are detailed in Table 1.

On the basis of the results, it is clear that the overall best performance considering MRR and Hits@K is shown by the msmarco-distilbert-base-tas-b model, while its memory requirements remained similar to the others. So, we decided to go ahead with this model as our bi-encoder. Since the zero-shot performance of the model is already pretty good and since fine-tuning the bi-encoder is computationally expensive, we decided not to fine-tune this model further on our dataset.

Since there are more TRUE questions than FALSE questions as specified earlier, we need to balance our dataset. To accomplish this, the bi-encoder model is used to generate negative samples that are added to the original dataset. The bi-encoder model retrieves the top 15 passages for each query in the training set, and for each query, the top passage that is not the correct answer is selected as the negative sample. By doing this, we generate additional negative samples and augment our dataset.

5 CROSS-ENCODER

The top 15 passages returned by the bi-encoder are now input into the cross-encoder model. The cross-encoder uses both the query and passage simultaneously, unlike the bi-encoder model, which takes queries and passages independently as input. This makes its evaluation and training more complex.

The cross-encoder combines the input and output sequences into a single sequence, usually with a special separator token between them, which is then encoded. Next, a similarity score is computed by passing the encoded vector representation through fully connected layers. On the basis of the similarity scores obtained from the cross-encoder, the 15 passages are re-ranked, and only the top 5 passages with the highest scores are propagated ahead to the QA model. Figure 5 depicts the working of the model.

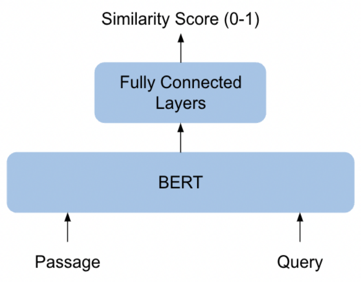


Figure 5: Cross-encoder working

· Model Name	MRR@5	MRR@10	MRR@15	Hits@5 (%)	Hits@10 (%)	Hits@15	Memory (MB)
msmarco-distilbert-base-tas-b	0.90808	0.91515	0.91565	98.181	98.181	98.787	192.423
all-mpnet-base-v2	0.77454	0.77768	0.77650	90.303	94.545	96.969	192.421
all-distilroberta-v1	0.84151	0.84727	0.84821	92.121	96.969	98.181	192.421
all-MiniLM-L12-v2	0.76545	0.76555	0.76795	90.303	92.727	94.545	96.210
multi-qa-distilbert-cos-v1	0.88181	0.88646	0.88646	98.181	98.787	98.787	192.421

Table 1: Comparison of zero-shot bi-encoder models

Model name	Hits@1	Hits@2	Hits@3	Hits@4	Hits@5	MRR	Inference time per query (s)	Memory
ms-marco-TinyBERT-L-2-v2	0.83529	0.90588	0.91764	0.92941	0.93333	0.87	0.12	16.739
ms-marco-Mini-L-2-v2	0.82745	0.89019	0.90588	0.92941	0.93333	0.87	0.59	59.577
ms-marco-Mini-LM-L-4-v2	0.84705	0.90980	0.92156	0.92941	0.93333	0.88	1.21	73.115
ms-marco-MiniLM-L-6-v2	0.85490	0.90588	0.91372	0.92941	0.93333	0.88	1.78	86.653
ms-marco-MiniLM-L-12-v2	0.85490	0.89803	0.92540	0.92940	0.93333	0.88	3.44	127.268

Table 2: Comparison of zero-shot cross-encoder models

Model name	Hits@1	Hits@2	Hits@3	Hits@4	Hits@5	MRR	Inference time per query (s)	Memory
ms-marco-TinyBERT-L-2-v2	0.85490	0.91372	0.93333	0.93333	0.93333	0.89	0.12	16.739

Table 3: Trained cross-encoder model

Similar to the bi-encoder, we carry out zero-shot evaluations of multiple cross-encoders in order to decide which model to use. These models are publicly available at https://www.sbert.net/. We compare the performance of these models on the basis of their MRR, Hits@K, memory requirements as well as inference time per query. The results are summarized in Table 2. Based on the results, it is evident that the performance of the ms-marco-TinyBERT-L-2-v2 model is superior when considering trade-offs between performance, model size, and inference time. Hence, we decide to use this model in our pipeline.

To train the cross encoder, a query, positive passage (ground truth), and negative passages are required. There are two ways for obtaining the negative passages for this task. The naive approach is to randomly sample passages other than the ground truth passage and label them as negative passages. The approach that we used in our pipeline involved the mining of hard negatives. In this approach, we first let the bi-encoder retrieve the top K passages from the entire set of passages. Among these K passages, we take the top N passages as the negative passages. In this way, the cross-encoder learns to differentiate between two highly semantically similar passages. Thus, we use the ground truth as the positive passage and the top 4 (excluding the ground truth) passages out of the 15 passages retrieved from msmarco-distilbert-base-tas-b as the hard negatives for training the cross-encoder.

The model is trained on our dataset using Localized Contrastive Estimation [15] (LCE) loss. Say q is the query and d represents the set of documents retrieved for the query q. Here, d^+ represents the ground truth document for query q and the score is the scalar output obtained from the cross-encoder. The loss is given by the following formula:

$$\mathcal{L}_{\text{LCE}} = -\log\left(\frac{\exp\left(\text{score}(q, d^+)\right)}{\sum_i \exp\left(\text{score}(q, d^{(i)})\right)}\right) \tag{1}$$

The performance of the model after training is depicted in Table 3.

6 QA MODEL

The final step in the pipeline is to extract the answer string from each of the top 5 passages obtained from the cross-encoder. We use pre-trained transformer-based models [16, 17] for this task. These models consist of language models pre-trained with the masked language modelling objective with a classification layer on top. They first take the input passage and query and convert it into embeddings. They then use a token classification approach to identify the answer to the query. The model predicts the probability of each token in the context being the start and end of the answer. The start and end tokens with the highest probabilities are selected as the answer span, which is extracted from the passage and then returned as the output along with a confidence score. The answer string, passage, and corresponding paper with the highest confidence score is returned to the user, provided that its score is above a pre-defined threshold. If the highest score is below this threshold, "answer not possible" is returned to the user.

We evaluate the zero-shot performance of multiple pre-trained QA models on the basis of the following metrics:

- F1 Score: The F1 score is the harmonic mean of precision and recall.

$$F1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{2}$$

Model name	F1 Score	Inference time per question on CPU (in s)
distilbert-base-cased-distilled-squad	0.54045	0.38
electra-small	0.51303	0.01
bert-large-uncased-whole-word-masking-squad	0.54782	>1.5
roberta-base-squad2	0.55513	0.85
minilm-uncased-squad2	0.49726	0.20
tinyroberta-squad2	0.48801	0.31

Table 4: Comparison of zero-shot QA models

Model name	F1 Score	Inference time per question on CPU (in s)
roberta-base-squad2	0.80510	0.85

Table 5: Trained QA model

In this context, precision is defined as the ratio of the number of shared words to the total number of words in the prediction, and recall is defined as the ratio of the number of shared words to the total number of words in the ground truth.

- Exact Match: Exact match is defined as the number of exact matches between ground truth answers and predictions.

The results of the zero-shot evaluations are detailed in Table 4. Considering trade-offs in F1 score and inference time, we decide to fine-tune the roberta-base-squad2 model for our purpose. The model performance after training is depicted in Table 5.

7 DISCUSSION AND CONCLUSION

The final pipeline consists of the following steps in our solution: Initially, we retrieve the top 15 passages using bi-encoder embeddings. After this, we retrieve the top 5 passages after re-ranking using the fine-tuned cross-encoder. On these 5 passages, the roberta-base model is applied individually to retrieve the answers and estimate the confidence scores. We then chose the answer string corresponding to the highest obtained score as our final answer string, its passage ID as the final passage ID, and the corresponding paper. This holds only if the best score is above the pre-defined threshold, if this fails, it implies that none of the passages answer the query and thus we return -1 as the passage ID and "answer not possible" as the answer.

In the future, we may explore implementation of additional features such as query reuse, and model quantization to speed up inference. We may also include a table information extraction step in the pipeline, to improve the results in question answering from tables. The models in our pipeline have been fine-tuned on a material science corpus however, it is possible to extend its use to other domains as well. This makes our pipeline a versatile tool that can be adapted to various fields, with the potential to improve information retrieval and answer extraction for a range of subjects.

8 ACKNOWLEDGEMENTS

This project was undertaken in fulfillment of the requirements for the ME 793 course at IIT Bombay. We would like to sincerely thank Professor Alankar Alankar of the Department of Mechanical Engineering, for his invaluable guidance and support throughout the research process.

REFERENCES

[1] Zach Jensen, Edward Kim, Soonhyoung Kwon, Terry Z. H. Gani, Yuriy Román-Leshkov, Manuel Moliner, Avelino Corma, and Elsa Olivetti. 2019. A machine learning approach to zeolite synthesis enabled by automatic literature data extraction. *ACS Central Science*, 5, 5, (May 2019), 892–899. DOI: 10.1021/acscent sci.9b00193.

[2] Pranav Shetty and Rampi Ramprasad. 2021. Automated knowledge extraction from polymer literature using natural language processing. *iScience*, 24, 1, 101922. DOI: https://doi.org/10.1016/j.isci.2020.101922.

[3] Vineeth Venugopal, Sourav Sahoo, Mohd Zaki, Manish Agarwal, Nitya Nand Gosvami, and N. M. Anoop Krishnan. 2021. Looking through glass: knowledge discovery from materials science literature using natural language processing. *Patterns*, 2, 7, 100290. DOI: https://doi.org/10.1016/j.patter.2021.100290.

[4] Tanishq Gupta, Mohd Zaki, N. M. Anoop Krishnan, and Mausam. 2021. Matscibert: a materials domain language model for text mining and information extraction. (2021). arXiv: 2109.15290 [cs.CL].

[5] Siddhartha R. Jonnalagadda, Pawan Goyal, and Mark D. Huffman. 2015. Automating data extraction in systematic reviews: a systematic review. *Systematic Reviews*, 4, 1, (June 2015), 78. DOI: 10.1186/s13643-015-0066-7.

[6] Olga Kononova, Tanjin He, Haoyan Huo, Amalie Trewartha, Elsa A. Olivetti, and Gerbrand Ceder. 2021. Opportunities and challenges of text mining in materials research. *iScience*, 24, 3, 102155. DOI: https://doi.org/10.1016/j.isci.202 1.102155.

[7] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. (2020). arXiv: 1910.10683 [cs.LG].

[8] Marcin Junczys-Dowmunt et al. 2018. Marian: fast neural machine translation in c++. (2018). arXiv: 1804.00344 [cs.CL].

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: pre-training of deep bidirectional transformers for language understanding. (2019). arXiv: 1810.04805 [cs.CL].

[10] Payal Bajaj et al. 2018. Ms marco: a human generated machine reading comprehension dataset. (2018). arXiv: 1611.09268 [cs.CL].

[11] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: masked and permuted pre-training for language understanding. (2020). arXiv: 2004.09297 [cs.CL].

[12] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. (2020). arXiv: 1910.01108 [cs.CL].

[13] Yinhan Liu et al. 2019. Roberta: a robustly optimized bert pretraining approach. (2019). arXiv: 1907.11692 [cs.CL].

[14] Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: sentence embeddings using siamese bert-networks. (Nov. 2019). https://arxiv.org/abs/1908.10084.

[15] Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021. Rethink training of bert rerankers in multi-stage retrieval pipeline. (2021). arXiv: 2101.08751 [cs.IR].

[16] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: deep self-attention distillation for task-agnostic compression of pre-trained transformers. (2020). arXiv: 2002.10957 [cs.CL].

[17] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: pre-training text encoders as discriminators rather than generators. (2020). arXiv: 2003.10555 [cs.CL].