Multi-Retriever Closed Domain Question Answering

[a]Anish Ghule, [a]Kamlesh Tiwari, [a]Ashutosh Bhatia, [b]Ashish Agarwal
[a] Dept. of CSIS, Birla Institute of Technology and Science Pilani, Jhunjhunu-333031, Rajasthan, INDIA
[b] NPBridge, Kiran Arcade 651, 13 Cross, Road-27, Sector-1, HSR Layout, Bengaluru, Karnataka 560102
{f20200129, kamlesh.tiwari, ashutosh.bhatia}@pilani.bits-pilani.ac.in, ashish@npbridge.com

Abstract

Objective of this research work is to build a novel system for closed domain question answering specifically for educational queries. The primary goal in Closed-Domain Question Answering (QA) is to extract answers to the query posed by user, within a specific domain. In this paper, we propose an end-to-end system for closed domain question-answering, based on two different embedding representation - sparse and dense, in the multi-retriever node. We also introduce a ranker, to rerank the retrieved documents from the multi-retriever node. We test our proposed system on five publically-available closed-domain datasets: European Law (Legal), Immune System (Medical), Pharmacy (Pharmaceutical), Nikolas Tesla (Person), and CovidQA (Biomedical) against six other QA approaches. Results show that the proposed QA system outperforms the existing approaches, based on the evaluation using standard metrics of Exact Match and F1-score. Moreover, in the case of CovidQA dataset, comprising of text which is complicated, our QA System shows a considerable improvement (increase in F1-Score). The proposed QA System can be utilized for building Question-Answering agents specialized in a particular domain, which can aid educational experience.

1. Introduction

Question Answering (QA) refers to the form of information retrieval for answering the posed question in human natural language from a pre-structured datastore, which may rely on a small corpus of data or large-scale knowledge base. Closed-domain QA utilizes the specifics of domain, such as domain-specific knowledge [17], for a small corpus of data. Open-domain QA involves answering generic questions which are not necessarily restricted to a specific domain of knowledge, and are often on large-scale knowledge base [3].

Existing QA sytems can be categorized into textualQA and knowledge-base (KB-QA), based upon the information source for answering the questions. TextualQA mines answer from unstructured documents [2], while KB-QA leverages a pre-defined structure [18]. Machine-reading comprehension (MRC) [19] in TextualQA involves reading and comprehension of specified context for question-answering. A system for closed-domain QA is proposed in [9] for educational queries in which the extracted keywords from the closed-domain education were stored in an index term dictionary. For finding the answer, Part-of-speech (POS) tagging was applied to all the filtered documents. A closed-domain QA framework based on rule-based classification and similarity-measure for "Hyderabad Tourism" is proposed in [1]. Covid-19 open research dataset (CORD-19) is used to create a closed-domain QA system [6] by fine-tuning Covid-related information. Question answering based on Chinese disease knowledge base is proposed in [17].

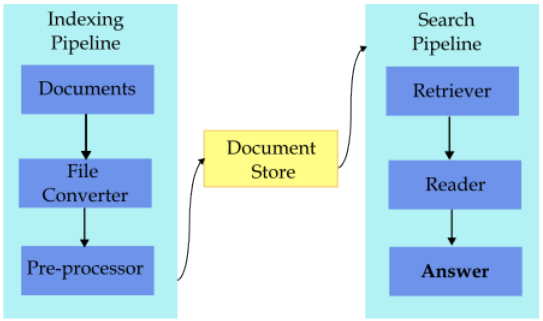

Figure 1: Traditional Question-Answering Pipeline consisting of and indexing pipeline and search pipeline. Indexing pipeline is used to create the document store, which is queried by the Search pipeline

Novelty: In traditional Question-Answering pipelines, there is a retriever-reader architecture, as shown in Figure 1. In the Document Retrieval stage, the system searches for documents relevant to the question, with the generated

search queries using existing IR techniques like TF-IDF. In the Answer Extraction stage, the final answer is extracted from related documents received from the retriever.

process[1] innovation of combining answers from multiple document stores based on different representation of the word embedding. The retriever module consists of multi-retrievers based on two different representation of embedding, sparse and dense. A *sparse* embedding retriever using ElasticSearch is used to search relevant documents for the posed query. Along with it, a *dense* embedding retriever - DensePassage Retriever [7] is used to retrieve the documents relevant to the posed user query.

By using the multi-retriever-reader QA system, we are able to improve the results that are retrived for a query posed by the user. A sparse embedding retriever is computationally less expensive, which comes at the tradeoff of substantially reduced retrieval effectiveness. The documents retrieved are less effective when the semantic gap in document store and user query is more. The Dense-embedding retriever is more effective in understanding the semantics of the posed query and documents. This compensates for the semantic gap in the sparse-retriever between query and documents with knowledge closed under a specific domain. The re-ranker component of our proposed QA system re-ranks the documents by calculating the semantic score using cross-encoder, which ensures accuracy in the retrived documents. Thus, our proposed system is capable of retrieving accurate results and is comparatively more effective than existing question answering systems in closed domain, which reflects from the standard evaluation metric results of Exact Match and F1-Score..

Rest of the paper is organized as below. Section 2 summarizes briefly the related work for closed-domain question answering. In Section 3, we describe each individual component in our proposed system - Sparse Embedding Retriever in 3.2.1, Dense Embedding Retriever in 3.2.2, followed by the Reader node explained in Section 3.3. The results of our work are elaborated in Section 4.

2. Related Work

Work related to open-domain question answering, is shown by [16], which has demonstrated the use of Wikipedia as a rich knowledge source in a question answering system with multiple answer matching modules, which is based on different types of semi-structured knowledge sources of Wikipedia. In order to retrieve an answer, the best answer a for a given question q is selected, such that the product of scores for question analysis, answer matching, and document retrieval are maximized. Maximize $SQ(r|q) \cdot SD(d|r) \cdot SA(M)(a|q,r,d)$ where r, a, d are question analysis result, answer candidate and retrieved document, respectively with the normalized scores (0-1). The primary objective of this work was to find a way to use the existing semi-structured and large-scale Wikipedia database as a knowledge source for a QA system, without the need to develop an expensive knowledge base.

A closed-domain QA system for user queries related to education is proposed by Lende and Raghuwanshi [9]. This method stores the extracted keywords in an index-term dictionary, and POS tagging is applied to all the filtered documents for finding the suitable answer. Badugu and Renganathan [1] propose a framework based on closed domain QA, which utilizes a rule-based classification approach and similiarity measure, specifically restricted to Hyderabad Tourism. The Jaccard similarity score was used to arrive at the answer which came closest to the user's question. They have additionally used "WordNet" in order to obtain the suitable response, which is dependent on syntactic and semantic similarities.

The authors of the paper [4] suggested a template representation model, the purpose of which is to map the given question onto the existing template question. To accomplish this, the entity under consideration will be substituted with its concept. It is accomplished using a method that is referred to as conceptualization. With the help of templates, it is possible to break down a complicated question into a series of questions, each of which corresponds to a different predicate.

There are two categories that can be used to classify reading strategies depending on whether the documents that were retrieved are handled individually or jointly for the purpose of answer extraction. A reading comprehension approach is used in various QA models. Through the utilisation of BERT Reader, The Dense Passage Retrieval [7] algorithm calculates the probability that a passage contains the answer to the posed question, as well as the likelihood that the token represents both the start and the end of an answer span.

Featuers such as POS, named entities (NE), and term frequencies (TF), are extracted from the context in DrQA [3]. The span of the answer is predicted by multilayer Bi-LSTM based upon the inputs, the question, and the paragraphs. The final average of the answer score calculated across paragraphs is using the $argmax$ function.

3. Proposed System

In this section, we have going to explain the proposed system in detail. The system has five broad components as shown in the block diagram Figure 2.

We propose an end-to-end QA System, consisting of the Sparse Embedding Retriever and Dense Embedding Retriever, which will retrieve the set of relevant documents from the pre-structured document store. The retrieved docu-

[1]Hybrid Sparse and Dense Embedding Retriever, Reader, and Re-ranker

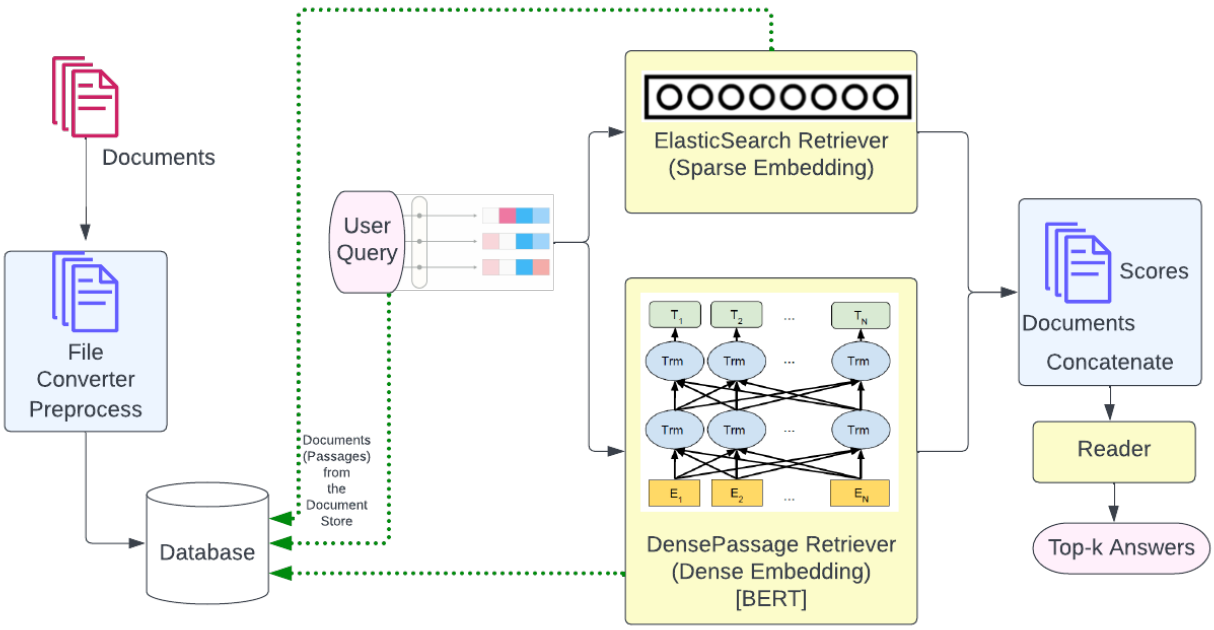

Figure 2: Block diagram of proposed QA System demonstrating that the sparse and dense embedding retriever results (set of relevant documents, along with confidence score) are concatenated. An extractive reader is used to extract the top-k answers, where k denotes the rank of the answer.

ments will be re-ranked by the Ranker Node. The extractive reader utilizes these documents and yields the answer.

3.1. Database Creation

We have used the proposed QA System for two domains, NLP (LectureBank) and Biomedical (CovidQA). We create the vector document store containing contexts in information-rich vector representation.

The LectureBank [11] database is used for curation of the dataset in NLP domain. LectureBank contains links of wikipedia articles for various topics in NLP. The Webpage documents pointed by these links are extracted to create a structured NLP dataset. In order to convert the unstructure webpage to a structured format, we have used the beautifulsoup4 library. The parsed content is stored it in a text file, which is pre-processed before storing in the document store. The pre-processing involves cleaning and splitting the text from Wikipedia articles. The header, footer, and semi-structured data like tables, boxes, lists is cleaned, and split by length of 200 words *passages* of . The CovidQA Dataset consists of contexts, which we first individually write in text files. These files are pre-processed by cleaning whitespaces, and splitting in length of 200 words, which form the basic unit for retriever and then stored in the document store.

3.2. Retriever

It is the module that takes the posed query and uses the document store to find the set of relevant documents, based on the similarity between the query and document. A single retriever node yields the top ranked potential documents from the document store. We propose to use multi-retrievers in our QA System, based on two different representation of embedding - *sparse* and *dense*. The document-score pairs are concatenated from the multi-retrievers as illustrated in figure 2. Each of the retrievers is explained below:

3.2.1 Spare Embedding Retriever

The sparse embedding retriever uses the sparse, high-dimensional, bag-of-words representation for ranking the document based upon the posed query. We use the ESRetriever, which is based on BM25 [15] similarity algorithm for ranking documents from the document store.

Suppose the posed user query is Q, comprising of $\{q_1, q_2, ..., q_n\}$ tokens, where n is the number of tokens, and response document is D. The response document have a length $|D|$, and the average document length for all indexed response documents be D_{avg}. The hyperparameter k helps determine the term frequency saturation characteristics, or in other words, constrains the effect that a single query term has, and is initialized with the default value of 1.2. b is also

a free variable initialized to the default value of 0.75, which penalizes long documents. Relevant responses are ranked using term frequency $f(q_i, D)$ and inverse document frequency $IDF(q_i)$. The BM25 score for sparse embedding denoted by S_s is determined using:

$$S_s(q, D) = \sum_{i=1}^{n} IDF(q_i) \cdot \frac{f(q_i, D) \cdot (k + 1)}{f(q_i, D) + k \cdot (1 - b + b \cdot \frac{|D|}{D_{avg}})}$$

Due to the computationally less intensive technique, sparse retriever is low-cost. However, it has substantially lower retrieval effectiveness. Sparse retriever are less effective when there is a semantic gap in the query and database.

3.2.2 Dense Embedding Retriever

The dense embedding retriever utilizes dense, low-dimensional representation for ranking documents based on posed user query, yielding a response which is not too biased towards a specific term. Dense embedding representation in information retrieval works efficiently even when there is a semantic gap between posed query and the document store in the latent space.

The standard BERT pretrained model [5] and dual-encoder architecture is leveraged in the DPRRetriever [7]. It works upon two unique BERT Encoder Models. One of those models, E_P, encodes passages of text into an encoded *passage* vector. The other model, E_Q, maps a question into an encoded *question* vector. By training the two models to output the same vector, the context encoder and question encoder learns to output very similar vectors for related question-context pairs. The similarity between the passage P and query q is defined using:

$$similarity(q, P) = E_Q(q)^T E_P(P)$$

The passage encoder E_P is implemented using BERT network. The dimensionality is 768, as the output is considered at [CLS] token. During inferencing, the passage encoder E_P is applied to all passages and indexed using ElasticSearch. For the posed query q, the embedding $v_q = E_Q(q)$ is derived, and relevant documents with embeddings closest to v_q are retrieved.

Dense embedding retriever is substantially effective in comparison to sparse retriver, but is computationally intensive and expensive. The dense representations of the query and document is learned. Bi-encoder is also advantageous in terms of linear time performance for large document store also.

3.3. Reader

Reader is the module which is inputted with the contexts returned by the vector data store and retriever module. It outputs the predicted answer for the posed user query, by returning a span (start and end tokens) of predicted answer.

The goal is to extract an answer from the obtained response documents. We use the pretrained model of BioBERT [8] base cased, and fine tune on Deepset CovidQA, which is a small subset of CovidQA [12]. BioBERT (Bidirectional Encoder Representations from Transformers for Biomedical Text Mining) uses the pretrained BERT on SQuAD, and trains on large scale biomedical corpora consisting of approximately 21 billion words, of which 18 billion words are from PubMed Abstracts and PMC Full-Text Articles. The cased model is more efficient in Biomedical contexts.

4. Results

4.1. Databases

We have used two databases, namely, LectureBank (NLP) and CovidQA (Biomedical). The LectureBank [11] database is used for curation of the dataset in NLP domain

LectureBank dataset
ID
Topic
Wikipedia page URL

The webpages pointed by the wikipedia links are scraped, parsed using beautifulsoup4 library and written in text files. This text is cleaned by removing header, footer, and semi-structured data like tables, boxes, lists, further split into passages of 200 words before writing to the document store.

In order to compare with other models, we have used the CovidQA [12] dataset. The CovidQA dataset consists of 2019 question-answer pairs annotated by volunteer biomedical experts on scientific articles related to COVID-19.

4.2. Experimental Setting

As shown in the block diagram (Figure. 2), we setup the document store, retriever module consisting of sparse embedding and dense embedding retriever, and the reader module for extractive QA.

The embedding dimension of document store is set to 768, which is the same as the embedding dimension of the model. The dimensionality is 768, as the output is considered at [CLS] token.

The sparse embedding retriver used is ESRetriver, based upon the BM25 ranking function. The free variables b and k are initialized with a value of 0.75 and 1.2 respectively in the score function. DPRRetriever, based on a bi-encoder (E_P and E_Q) is setup as the dense embedding retriver. The top-k passages are retrived as response documents from document store, based upon the posed user query for both the retrivers, where we have set the hyperparameter k to 5 (top-5 documents ranked basis the score from BM25).

Relevant response documents obtained from the retriver module are concatenated into a single pool, which serves as the input for the Extractive Reader. The answer span is between the start and end tokens, which is the predicted answer by the reader. The top-k answer responses are answers for the posed query.

Summarizing the experimental settings in table below:

embedding_dim	768
Retriever	ElasticSearchRetriever DensePassageRetriever
Reader	FARMReader
top_k (ESRetriever)	5
top_k (DPRRetriever)	5
top_k (QAReader)	2

4.3. Evaluation Parameters

To analyze the performance of our proposed system we have used four standard measures $viz.$ Precision , Recall, F1-score, and Exact-Match (EM). A brief definition of these measures as follows:

Suppose that D denotes the number of relevant documents retrieved, from a total of R actual relevant documents, and T denote the total number of documents retrieved.

Precision Precision is a measure of the number of positive predictions made being correct. Mathematically, it is the ratio of the number of relevant documents retrieved to the total number of documents retrieved. $Precision = \frac{D}{T}$

Recall Recall measures how many times the correct document was among the retrieved documents over a set of queries. For a single query, the output is binary: either the correct document is contained in the selection, or it is not. Over the entire data store, recall score is in the range of zero (no query retrieved the right document) and one (all queries retrieved the right documents). $Recall = \frac{D}{R}$

F1-Score The top-1 F1 score represents the average overlap between first predicted answer and correct answer. Mathematically, it is expressed as the Harmonic Mean of Precision and Recall.

$$F1 - score = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

Exact Match For each question+answer pair, if the characters of the predicted answer exactly match the characters of the True Answer(s), EM = 1, otherwise EM = 0. The Exact Match (EM) score is a strict all-or-nothing metric, or in other words, a single different character results in a score of 0.

4.4. Observations

We conducted our experiment as per the experimental settings mentioned in Section 4.2, for our proposed QA system. In the Table 1 we have discussed state of the art, and shown a comparison of our method with the existing methods. We adopt two metrics including Exact Match (EM) and F1 scores to evaluate our model. The EM score determines the percentage of predictions that perfectly match the ground truth answer, and the F1 score demonstrates the average overlap between the prediction and the ground truth answer. We obtained an EM score of 66.0% and F1 score of 81.6% on the CovidQA Dataset [12]. Compared to the QANet model proposed in [13], the proposed model outperforms largely with a difference of 53.6% in the Exact Match (EM) Score, and 38% in the F1-score. In comparison to the OCovidQA [10], we notice an improvement by 26.9% in EM Score and 9.6% in the F1 Score.

Model	**EM**	**F1**
QANet [13]	12.4	43.6
ZCovidQA [14]	25.9	59.5
OCovidQA [10]	39.1	72.0
Proposed	48.5	66.2
Proposed (BioBERT)	**66.0**	**81.6**

Table 1: Comparison of the proposed system with state of the art methods on CovidQA Dataset consisting of 2019 QA pairs.

- QANet [13], is an MRC model for open-domain QA based on convolutions, global self-attention, and the GLoVe language model.
- ZCovid-QA [14], employed RoBERTa fine-tuned on the SQuAD and QuAC datasets for zero-shot evaluation on the COVID-QA dataset for Covid-19 QA.
- OCovid-QA [10] utilized a variant of BioBERT fine-tuned on the SQuAD2.0 and COVID-QA datasets for Covid-19 QA.

4.5. Ablation Study

For closed-domain QA, we consider the single-retriever nodes, similar to the traditional QA systems (Figure 1). Here, the single node for the retriever is either a Sparse-embedding retriever, such as ElasticSearch Retriever, or a Dense-embedding retriever, such as EmbeddingRetriever or DensePassageRetriever [7].

We investigate the effect of each of the individual retriever, and that of the ranker node, to understand the overall role of each node in the overall system. In Table 3, the analysis of the EM, F1, SAS scores, as individual nodes is shown in the first three rows. The performance is reduced significantly by using only the Dense-embedding Retriever like DPR or EmbeddingRetriever. The hybrid retriver shown in rows 4 and 5, is able to capture the semantic

Sr no.	Query	Golden answer	Predicted ans	EM	F1
1	What does alternative splicing of the DC-SIGNR gene in the placenta produce?	both membrane-bound and soluble repertoires	both membrane-bound and soluble repertoires	1	1
2	What is the percentage of Mother to Child Transmission of HIV-1, when there is no intervention?	Without specific interventions, the rate of HIV-1 mother-to-child transmission (MTCT) is approximately 15-45%	15-45%	0	0.15
3	What regulates the antiviral activity of IFITM3?	S-palmitoylation on the protein	S-palmitoylation on the protein	1	1
4	How many nucleotides does bovine coronavirus contain?	30,847 nucleotides	30,847	0	0.67
5	How is exhaled breath condensate used in viral research?	the isolation of respiratory viruses	an inventive and novel method for the isolation of respiratory viruses	0	0.62

Table 2: The above is a representation of predicted answers to some questions from the CovidQA Dataset. The Golden Answer is the annotated answer by biomedical experts. EM is the exact match evaluation metric, and F is the F1-Score

Model	EM	F1	SAS
es	73.11	82	82.6
emb	64.29	72.9	73.88
dpr	42.44	51.62	57.15
es+emb	76.89	85.42	85.83
es+dpr	73.95	82.48	84.43
es+emb+ranker	**78.57**	**87.32**	**87.08**
es+dpr+ranker	73.95	82.48	84.43

Table 3: Ablation Study based on Hit@1 (top-k, k = 1). Here, es is the Sparse-embedding retriever. emb and dpr are Dense-embedding retrievers. The ablation shows the ranker component, which re-ranks the documents retriever from the previous node. Our proposed system uses the 'Sparse-retriever + Dense-retriever + Ranker'

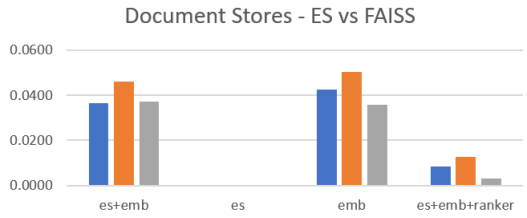

Figure 3: Negligible change is observed with change in Document Stores, indicating that the results obtained are not correlated with the type of Document Store being used.

relations between sentence and the posed question which increased the performance in EM, F1, SAS by 4% from es, and around 14%-35% from the results of the Dense-embedding retrievers.

We depict the graphical representation in Figure 4. We additionally conduct the experiment by using the FAISS Document Store, and ElasticSearch Document Store for storing the documents and observe that there is negligible (close to zero) difference in the scores obtained, as shown in Figure 3.

5. Conclusions

Traditional QA systems consist of a search pipeline for document retrival. The problem that is faced by such a system is that in closed-domain question answering, the knowledge is closed under a specific domain, and the semantic gap between posed user query and document store is not acknowledged by this system. Our proposed QA System consists of multi-retrivers based on Sparse and Dense Embedding representations, which has significantly improved the understanding of the model to domain-specific knowledge. The re-ranker node calculates the semantic-similarity score based on a cross-encoder and ensures that accurate documents are ranked, which is pipelined to the Reader node. We conclude that the proposed multi-retriever-reader QA system is effective in retriving an answer to the posed user query, based on the results obtained from experimentation. We get an EM Match accuracy of 66.0% and a F1-Score of 81.6% on the CovidQA dataset. As compared to the existing methods, an improvement of 17.5% to 27% EM score is observed, and increased F1-Score by 9.6% to 15.4%. We draw the conclusion that the proposed QA System is independent of the type of document store by experimenting with ElasticSearch and FAISS Document Stores. Domain-

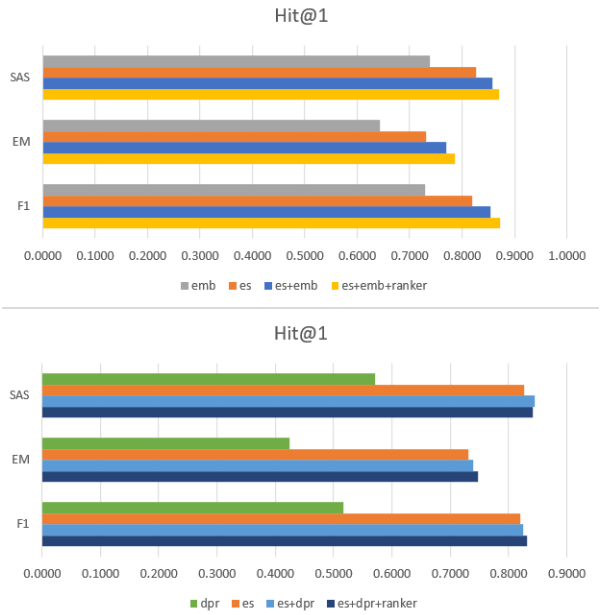

Figure 4: Comparative graphical representation of the nodes in terms of standard evaluation metrics EM (Exact Match), F1-Score, SAS (Semantic Answer Similarity) based on top-k documents retrieved, where k = 1. The figure shows that the proposed system 'Sparse-retriever + Dense-retriever + Ranker' results in better EM, F1, SAS, in comparison to the Single-retriever individual node.

specific knowledge is better acquired with a substantial improvement in the EM and F1 Score for our proposed QA System with the multi-retriever reader and ranker nodes.

References

[1] Srinivasu Badugu and Manivannan Renganathan. A study on different closed domain question answering approaches. *International Journal of Speech Technology*, 23, 06 2020.

[2] Yang Bai and Daisy Zhe Wang. More than reading comprehension: A survey on datasets and metrics of textual question answering. *CoRR*, abs/2109.12264, 2021.

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

[4] Wanyun Cui, Yanghua Xiao, Haixun Wang, Yangqiu Song, Seung won Hwang, and Wei Wang. KBQA. *Proceedings of the VLDB Endowment*, 10(5):565–576, jan 2017.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[6] Deepthi Godavarthi and Mary Sowjanya A. Queries related to covid-19: a more effective retrieval through finetuned albert with bm25l question answering system. *World Journal of Engineering*, 2021.

[7] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. *CoRR*, abs/2004.04906, 2020.

[8] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

[9] Sweta P. Lende and Mukesh M. Raghuwanshi. Question answering system on education acts using nlp techniques. *2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare (Startup Conclave)*, pages 1–6, 2016.

[10] Sharon Levy, Kevin Mo, Wenhan Xiong, and William Yang Wang. Open-Domain question-Answering for COVID-19 and other emergent domains. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 259–266, Online and Punta Cana, Dominican Republic, November 2021.

[11] Irene Li, Alexander R. Fabbri, Robert R. Tung, and Dragomir R. Radev. What should I learn first: Introducing lecturebank for NLP education and prerequisite chain learning. *CoRR*, abs/1811.12181, 2018.

[12] Timo Möller, Anthony Reina, Raghavan Jayakumar, and Malte Pietsch. COVID-QA: A question answering dataset for COVID-19. In *Proceedings of the 1st Workshop on NLP for COVID-19 at ACL 2020*, Online, July 2020.

[13] Arantxa Otegi, Jon Ander Campos, Gorka Azkune, Aitor Soroa, and Eneko Agirre. Automatic evaluation vs. user preference in neural textual QuestionAnswering over COVID-19 scientific literature. In *Proceedings of the 1st Workshop on NLP for COVID-19 (Part 2) at EMNLP 2020*, Online, December 2020.

[14] Revanth Gangi Reddy, Bhavani Iyer, Md. Arafat Sultan, Rong Zhang, Avi Sil, Vittorio Castelli, Radu Florian, and Salim Roukos. End-to-end QA on COVID-19: domain adaptation with synthetic training. *CoRR*, abs/2012.01414, 2020.

[15] Stephen E Robertson and Steve Walker. Okapi/keenbow at trec-8. In *TREC*, volume 8, pages 151–162, 1999.

[16] Pum-Mo Ryu, Myung-Gil Jang, and Hyun-Ki Kim. Open domain question answering using wikipedia-based knowledge model. *Information Processing Management*, 50(5):683–692, 2014.

[17] Xuan Wang and Zhijun Wang. Question answering system based on diease knowledge base. In *2020 IEEE 11th International Conference on Software Engineering and Service Science (ICSESS)*, pages 351–354, 2020.

[18] Mo Yu, Wenpeng Yin, Kazi Hasan, Cicero Dos Santos, Bing Xiang, and Bowen Zhou. Improved neural relation detection for knowledge base question answering. pages 571–581, 01 2017.

[19] Chengchang Zeng, Shaobo Li, Qin Li, Jie Hu, and Jianjun Hu. A survey on machine reading comprehension: Tasks, evaluation metrics, and benchmark datasets. *CoRR*, abs/2006.11880, 2020.