Web-interface for Language Processing Systems Project Information Retrieval-Based Biomedical Question Answering System

Presented by:

Sushil Awale, Sara Mínguez Monedero and Pablo Robles de Zulueta

Language Technology Group
Department of Informatics
Universität Hamburg
29.09.2022

1 Introduction

Question Answering (QA) is branch of computer science that relies on information retrieval (IR) and natural language processing (NLP) to be able to create systems that can automatically respond to queries from humans. QA systems started at 1980's with the development Unix Consultant at U.C. Berkeley. It was able to answer questions about Unix operating system [1]. From this system to the ones that exist today, there has been a great advance. This great advance could lead to improvements in the medical field, helping to find solutions to medical problems [1], [2].

Nowadays, we live in a society in which information is very easily available. However, in certain areas, we are not able to differentiate between quality information and not. An example of this can be the medical field due to general lack of medical knowledge of the population. More precise answers should be provided to avoid this [1]. For this purpose QA systems based on PubMed articles could be created. In addition, even for health care workers, current systems provide them with the following long lists of articles rather than self-contained answers to specific questions, and many of these articles turn out to be irrelevant to specific questions due to the inevitable query ambiguity of open-domain search engines. This is time-consuming and delays the instant assistance required to solve the pressing needs of the patients [1], [2]. Situations like the pandemic, have shown that there can be a collapse of the medical system so having reliable QA could help in telecare and make patients feel safer even without the presence of a doctor [3].

2 System description

In this section, our Information Retrieval-Based Biomedical Question Answering System (IR-based QA system) will be presented. Figure 1 depicts the overall system architecture, and this section will go into additional depth about each of its components. In any IR-based QA system, the document reader and document retriever are the two key parts of the model [2]. In our scenario, text extraction was done in an offline fashion prior to document retrieval. This lead to the creation of a biomedical-text database, stored in ElasticSearch. Thus, the system has to connect to the database every time a query is done. The rest of the system functions online, with FAST API being required to connect the user interface (UI), from where a query is inputted. This information will reach the back-end where, first, the natural language (NL) query is be transformed in the query formulator. Then, the transformed version is used to retrieve relevant passages stored in the already created database. Finally, using most relevant pieces of information —context— and a BERT-based QA model—document reader — [2], the top 5 final responses are be provided to the user together with the corresponding context and additional metadata.

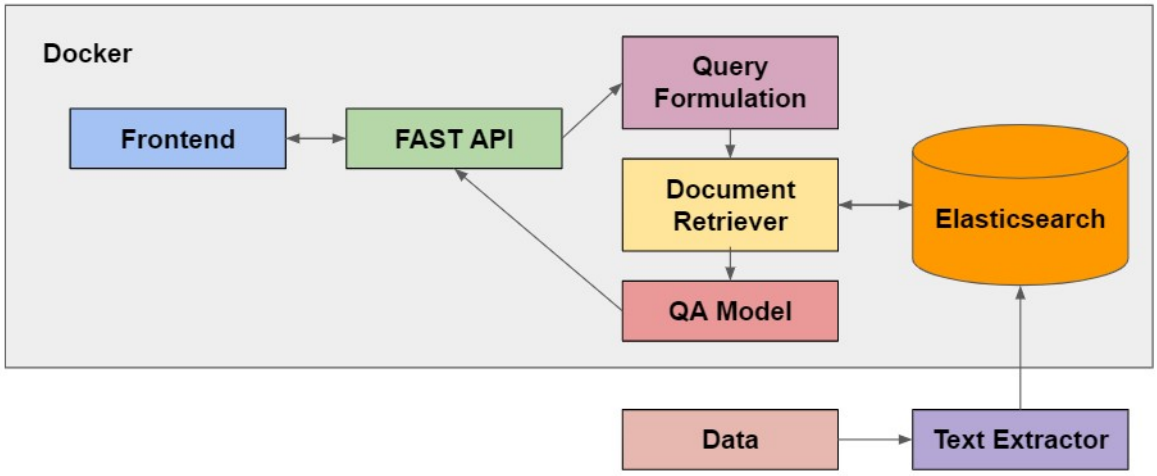

Figure 1: Overwiew of the architecture of our system and its components.

2.1 Data

Two different data sources have been used for the purposes of performing this project. In order to get the answers to the queries, it was necessary to retrieve high-quality scientific papers. They were obtained from the National Center for Biotechnology Information's (NCBI) PubMed Central (PMC), a repository of biomedical and life sciences papers. There are over 8.3 million articles in it [4]. Among the different digitisation formats, PMC offers different free full-text subsets [5]. In the one intended for bulk download, the text and metadata from articles or author manuscripts can be found [5]. Open Access articles are stored in compressed bulk packages, which group thousands of articles in XML or plain text formats. Depending on the purpose of the system, PubMed offers articles for commercial or non-commercial use, the later being the ones of election [4], [5]. However, due to space restrictions, around 173K articles were used for this project, which are stored as 6.9 million paragraphs/docs in Elasticsearch. Moreover, basic definitions are not often present in scientific papers as they are usually targeted at a more knowledgeable audience with prior notions of the subject matter. For instance, in an article dealing with bone cancer, definitions of specific terms such as *"what are osteocytes?"* —the main cell of bone tissue— will probably not be included. Thus, if a user were to ask the QA system *"what are osteocytes?"*, there would be a strong likelihood of not receiving an answer. This, for sure, could be extrapolated to other medical terms. Therefore, to overcome this limitation, definitions of biomedical concepts were included using the Biomedical Informatics Ontology System (BIOS) [6]. BIOS includes over 7 million biomedical concepts, which are matched to around 250 thousand definitions. It is worth noting that BIOS concepts are stored both in Chinese and English, whereas definitions are just stored in English.

Figure 2: Data sources from which the data was retrieved for the project. Images obtained from [4], [6].

2.2 Text Extractor

In order to create the biomedical database from where the information/context is retrieved, two text extractor modules were designed.

The first one is used to extract and pre-processes text from PMC bulk packages group and load it into Elasticsearch. As described in 2.1, research articles from PubMed were used. At the beginning, it was intended to automatically download documents periodically since PMC produces daily uploads [5]. However, due to the aforementioned space restrictions, we manually downloaded the September dump of *tar files* of PubMed Open Access Non-commercial from the official website. In particular, we download the research articles with PMC Id ranging from 000000000 to 002999999, which amounts to about 173 thousand research articles.

From the possible download options, the selected research articles were stored in XML format, which makes it easier to parse. Using the XML tags, paragraphs from the research articles are extracted. These are later stored as individual document in Elasticsearch along with the metadata of their corresponding research article. The extracted metadata includes title, author names, publication date and publication journal. For pre-processing, citations in the passages, which use square brackets — [ ] —, and HTML tags inside the passages were removed.

Regarding the biomedical concepts downloaded from BIOS —see Section 2.1—, it was essential to match the concepts with their definitions and remove concepts in other languages that were not English. In the pre-processing stage, the definitions which did not had associated concepts were removed. Then, since there are over 28 times more concepts than definitions, firstly, all the concepts in Chinese were removed, which lead to a 2 million reduction. Subsequently, since the relation between concepts and remaining definitions is surjective, meaning that all concepts have at least one definition and more than one concept can be matched to the same definition, repeated concepts were deleted. For instance, the terms *microglia*, *microglia cells* and *microglia cell* were matched to the same concept. In order to keep just one concept matched to each definition, the base concept —from which the rest of the derivations were made— was the only one kept. Finally, a total of 248.345 concepts with their corresponding definitions were acquired.

For document storage, Elasticsearch was used. Elasticsearch is a distributed search engine that indexes documents for fast-retreival [7]. We selected Elasticsearch due to its powerful text-based searching capabilities.

2.3 Query Formulation

The query formulator consists on 8 different submodules, whose aim is to extract the important keywords from the original question, which is inputted in natural language. Since document retrieval makes use of words —and not their vector form—, to select the most relevant information passages, the output of the query formulator will be a set of relevant words. For the different language processing related the Python libraries *spaCy*, together with *ScispaCy*, *RegEx*, *Contractions* and *Difflif* have been employed.

spaCy is an open-source library that was designed for Natural Language Processing production usage rather than teaching and research purposes —for example *NTLK*— [8]. NLP tasks are done by employing an already pretrained pipeline, which supports a variety of different functions. Some of the basic purposes for which spaCy can be applied range from parsing or part-of-speech (POS) tagging to named entity recognition (NER), dependency parsing or text categorisation [8]. Besides its outstanding speed and accuracy [8], spaCy was elected due to its integration with ScispaCy. The latter is a Python package that contains spaCy pipelines and models for processing biomedical text [9]. ScispaCy models have been finetuned utilising biomedical data to improve the processing of health-related scientific text. Precisely, the modified elements are the tokenizer, POS tagger and syntactic parser. Nonetheless, on top of the enhanced basic functions, some ScispaCy models, and thus pipelines, offer other useful components such as NER, Abbreviation Detector or Entity Linker among others [9].

It is worth noting that during the development of the QA system, specifically dependency parsing during the query formulator —see Figure 3—, it was observed that ScispaCy outcomes was not accurate with some sentences, even with simple ones. Because of this, both a spaCy — *en_core_web_sm* [8]— and ScispaCy — *en_core_sci_lg* [9]— pipelines were used for the different tasks involved in the query formulation.

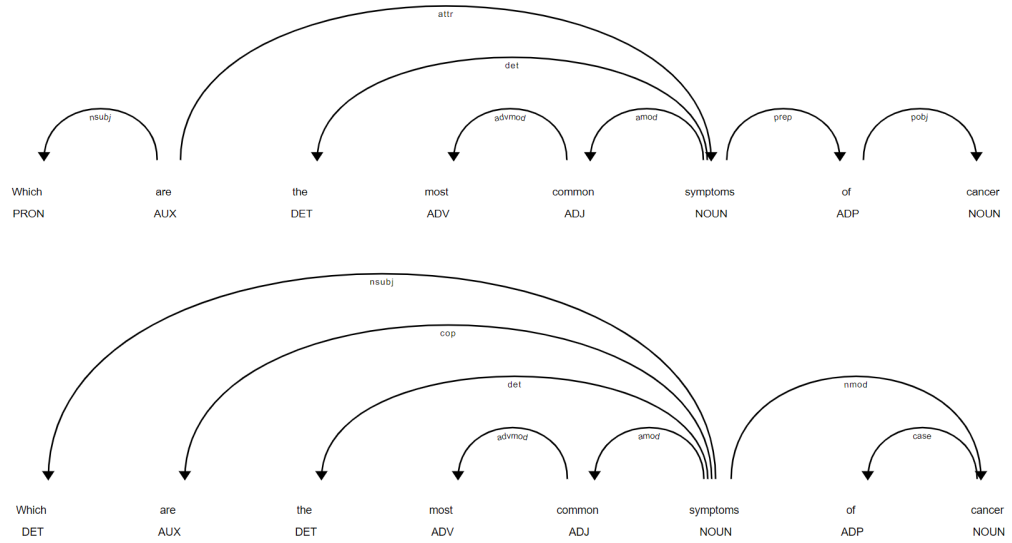

Figure 3: Dependency parsing of *"What are the most common symptoms of cancer?"*. Top shows a good example (*en_core_web_sm*) whereas bottom a bad one (*en_core_sci_lg*).

Before detailing the steps needed to obtain a set of relevant keywords from the natural language question, it should be mentioned that custom list of 235 stop words had to be tailored for this specific task. This is highly relevant since words such as not, mild, severe or little tend to be considered stop words. However, when describing a medical condition, a precise explanation of the symptoms is often crucial allow to track its development of a and even to unveil its cause. Since the model will work searching the information in papers with precise detailed, it was considered to create a shorter stop words list. Moreover, this problem can be detected in the following example: "I don't see through my right eye":

1. Custom stops: not see right eye.
2. Common stops: see right eye.

After having explained the different involved modules, now the different components —introduced as steps— of the query formulation will be explained. To track the transformation of the sentence done in the query formulator, two sentences and their transformations will be used as examples at the end of the enumeration in Table 1.

1. **String format sentence preparation**: First, if present, the contractions on the sentence will be expanded. Subsequently, the sentence is lower cased, and the different punctuation and special characters —except from parenthesis— are removed. Finally, if there is more than single spacing between the words, they are changed to single spaces.

2. **SpaCy Parsing**: The sentence is transformed into a spaCy document. In this format it is possible to extract all the semantical and lexical features from the text, crucial for the following steps.

3. **Medical entities extraction**: The prepared string format sentence is transformed into a spaCy document, but this time using the ScispaCy medical pipeline. Then, the medical entities are extracted. It should be noted that just medical entity extraction is performed rather than NER, where the noun class is considered (e.g., CELL, GENE, CHEMICAL, etc). Additionally, an abbreviation detector has been included. Its purpose is to detect contracted forms of a medical entity and substitute them with their expanded form.

4. **Noun chunk extraction**: Using the sentence transformed into a spaCy document in the second step, extract the noun chunks. This step is important because in some cases medical entities are just individual nouns and do not include adjetives or the rest of the nouns from the same noun group.

5. **Medical entity expansion**: In this step, the original medical entities —third step— will be compared against the noun chunks —fourth step—. The intention is to include possible noun modifiers (in this case adjectives) and to combine entities that might have been detected separately but actually are a unique noun group.

6. **Dependency parsing**: Until this point, the query formulator has not considered all the sentence's syntactic properties since it has just worked with the nouns —-in form of medical entities or noun chunks— and their adjectives. For this reason, the next step is to investigate the root of the sentence and the main subject. From those, they and their children and head will be included into the final sentence if they are not already part of the extended medical entities.

7. **Stop words removal**: A list of all possible candidates will be created combining the extended medical entities, conformed in steps 3, 4 and 5; together with the candidates from the dependency parsing. Then, using the custom list of stop words, these will be removed.

8. **Sentence construction**: Finally, preserving the original order of the sentence, candidates will be ordered to conform the query with the most important keywords.

	Sentence 1	Sentence 2
Original sentence	*"What's the most common clinical manifestation of Cytomegalovirus (CMV) infection in AIDS patients?"*	*I don't feel my hands, what can it be?*
String preparation	*What is the most common clinical manifestation of Cytomegalovirus (CMV) infection in AIDS patients?* *what is the most common clinical manifestation of cytomegalovirus (cmv) infection in aids patients*	*I do not feel my hands, what can it be?* *i do not feel my hands, what can it be*
SpaCy parsing	—	—
Medical entities	*'clinical', 'manifestation', 'cytomegalovirus (cmv) infection', 'patients', 'aids'*	hands
Noun chunks	*most common clinical manifestation', 'cytomegalovirus (cmv) infection', 'aids patients'*	hands
Entities expansion	*'most common clinical manifestation', 'aids patients', 'cytomegalovirus (cmv) infection'*	hands
Dependency parsing	*'what', 'is'*	'not', 'be', 'do', 'i', 'feel'
Stop words removal	*what* and *is* were removed	*be, do, i* were removed
Final query	*common clinical manifestation cytomegalovirus cmv infection aids patients*	*not feel hands*

Table 1: Tracking of the process of the sentence transformation and keywords extraction in the query formulation process.

2.4 Document Retrieval

Document Retrieval is an important part of the system as it determines which passages are fed into the question answering model. We devise a two-stage document retrieval process, as it can be observed in Figure 4.

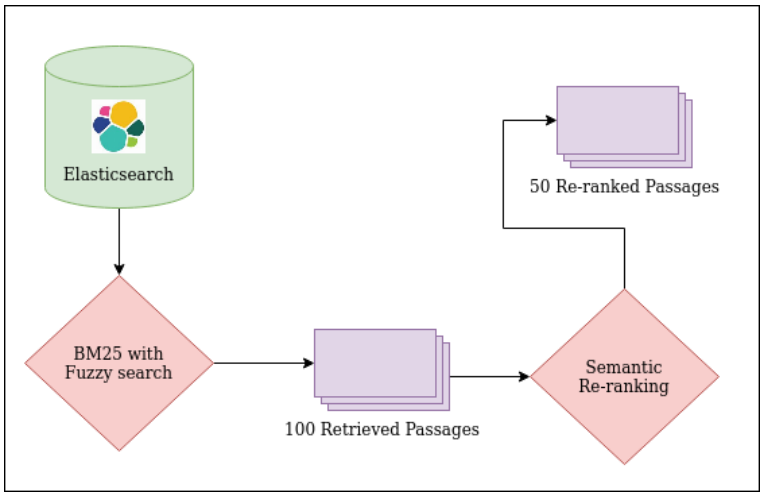

Figure 4: Document Retrieval Process

In the first phase, we retrieve the passages using text-based search. We use Elasticsearch's in-built Best Matching 25 (BM25) [1] ranking function for this purpose. BM25 is a bag-of-words retrieval process that weighs the occurence of query words in the passages. In addition, we also apply fuzzy search during the retrieval process with a prefix length of two which allows for users to make spelling mistakes up to two characters. We select the top 100 passages from this phase.

In the second phase, we re-rank the passages using semantic search. Firstly, we create embedding vectors of the 100 passages using MiniLM [10]. MiniLM is a passage model trained on the task of passage ranking on the MSMARCO (Microsoft Machine Reading Comprehension) dataset [2]. The dataset consists of 500 thousand real queries from Bing search accompanied with relevant passages from various domains. Next, we rank the passages based on cosine similarity score between the passage and the user's question. We select the top 50 passages from this phase which are fed into the question answering model.

2.5 Question Answering

The question answering model is able to, given a natural language query and a context from previously analyzed scientific papers, return the correct answer. For this purpose, reading comprehension algorithms are applied to the text segments obtained from the document retriever. Two different types of algorithms can be used: feature-based and neural-

[1] https://en.wikipedia.org/wiki/Okapi$_BM25$
[2] https://github.com/microsoft/MSMARCO-Passage-Ranking

based [1], [2], [11]. To extract features, feature-based answer extraction applies NLP models—e.g., named entity recognition (NER) or parts-of-speech tagging (POS)—and rule-based templates. Examples of relevant characteristics that will aid in determining the correct response include the type of response, the distance between the words in the query and the candidate response, or the quantity of matched keywords in the query. These features will then be used to determine in a supervised manner if the segment contains the answer or not. However, this kind of answer extraction has limitations, being more used nowadays neural-based models [1], [2].

Neural-based answer extraction relies on the assumption that the question and the answer are semantically similar. Therefore, these models will learn semantic embeddings for the question and the text passage. Based on these embeddings and using similarity functions, the final answer will be extracted [2], [11]. The neural networks that have better performance are the Seq2Seq models and the Transformers models. Pretraining large neural language models, such as Bidirectional Encoder Representations from Transformers (BERT), has led to impressive gains on many natural language processing (NLP) tasks. The BERT model was trained on Wikipedia and BookCorpus. Therefore, it would have low performance while working with biomedical texts [12].

To solve this problem, in 2019, BioBERT appeared [13]. This is a mixed-domain pretrained model based on the continual pretraining of a general-domain pretrained model. As it can be observed in Figure 5, it initializes with the standard BERT model, and it continues pretraining with PubMed articles. However, the problem of this is that the vocabulary is the same as the original BERT model, in this case the one generated from Wikipedia and BookCorpus [12], [13]. This is a major disadvantage for this approach, as the vocabulary is not representative of the target biomedical domain. Therefore, domain-specific pretraining from scratch is a better strategy for biomedical language model pretraining. PubMedBERT is directly trained on PubMed articles, having an in-domain vocabulary, leading to better results while working with biomedical texts (Figure 5) [14]. This is the major reason it was selected for this project.

Finally, task-specific tuning is needed to make this pretrained linguistic model capable of responding to the questions [13], [14]. The Stanford Question Answering Dataset (SQuAD) v.2 was utilized for this [15]. This is a dataset, composed of questions asked by crowdworkers on a set of Wikipedia articles. One of the advantages of this dataset over its predecessor is that the answer, in addition to being a segment of the passage, can be that no answer is provided. This is possible since it combines the 100,000 questions from SQuAD 1.1 with more than 50,000 unanswered questions adversely written by crowdworkers to resemble those that are answered. In this way the model should be able to not only answer the question well, but also not answer it if no possible answer in the context it is being provided [15].

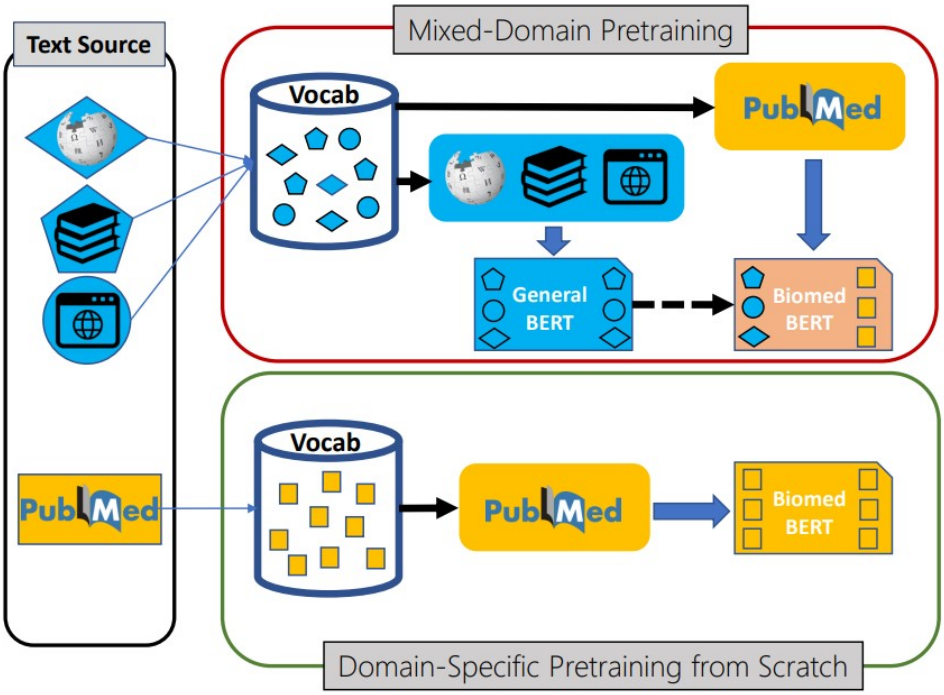

Figure 5: Two schemes for neural language model pretraining. Top: depicts the training for BioBERT model. Down: depicts the training for PubMedBERT model. Image obtained from [14].

The question answering model is given the 50 passages retrieved from the document retriever and it finds the answer for the user's questions in these passages. Not all passages contain answers, therefore, we disregard the passages that do not generate any answers. We only display five passages with answers to the user.

2.6 Deployment

The system deployment architecture is shown in Figure 6. We deploy the system using a Virtual Machine (VM) on Google Compute Engine. We utilize the $300 U.S. dollars given by Google Cloud Platform on sign up for using its services. The virtual machine has a 16-core Intel CPU with 100 GB of Hard Disk storage and runs Ubuntu 20.04 LTS.

The Elasticsearch instance and the application instance developed using FAST API are run as Docker containers. The user interface of the system is built using HTML and Bootstrap and is hosted in the VM as well.

We expose the system API to the public using NGROK [3] which is a programmable endpoint that creates a proxy for the *localhost*.

[3] https://ngrok.com/

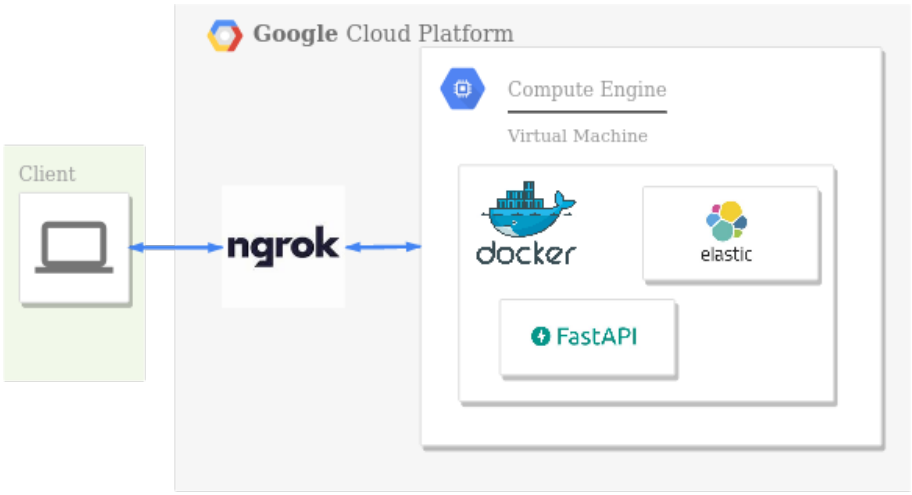

Figure 6: System Deployment Architecture

3 User Guide

In this section, we define how to run the system and how to use the system. We make the code public on GitHub [4].

3.1 Running the system

Prerequisite: Docker needs to be installed in the system

The following command starts the Elasticsearch instance and installs all dependencies for the FAST API web app, and starts the system.

- Run **docker-compose up –build**

The system can be accessed at http://0.0.0.0:8000. Figure 7 is the screenshot of the landing page. On this page you can see that 5 example questions appear. These questions belong to a set of 15 questions obtained from the BioAsq dataset[16]. This is a biomedical question answering dataset. Each time the landing page is accessed the set of questions will vary allowing the user to have a different list of suggested questions.

[4]https://github.com/awalesushil/biomed-qa

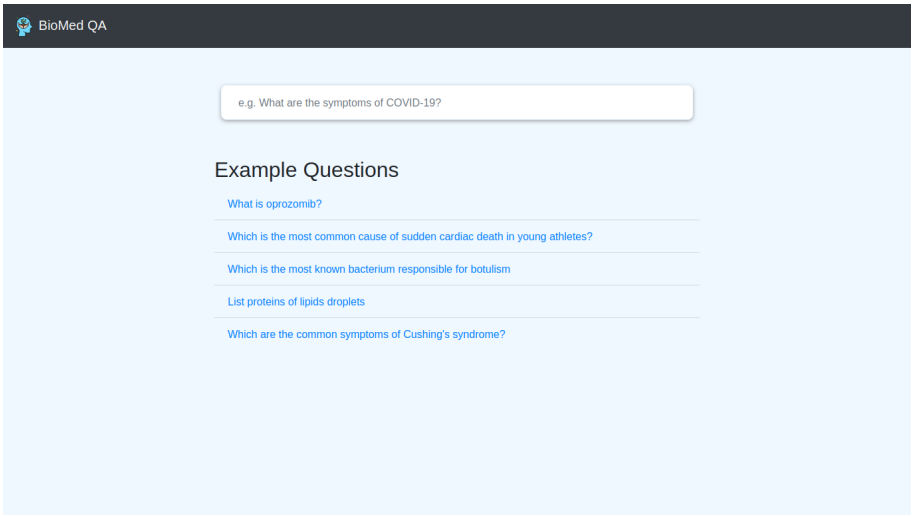

Figure 7: Landing page of the system

- Use $Ctrl + C$ to shutdown and then run **docker-compose down**

3.2 Loading the data

Prerequisite: Download the PubMed XML data dumps

The *loader_script.py* needs to run from outside the docker container since the data is not copied into docker context.

- Create a Python environment and install all dependencies.

python3 -m venv .env
source .env/bin/activate
pip3 install -r requirements.txt

- Load the data by running *loader_script.py*.

3.3 Asking question

The user can now directly type in a medical question in natural language text format on the search input section. The system will display up to five passages with answers to the question highlighted in a separate section as shown in Figure 8

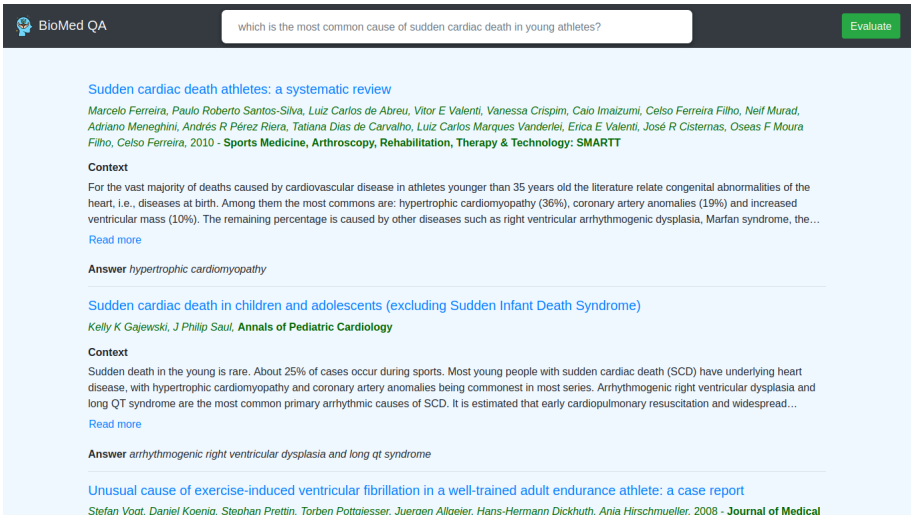

Figure 8: Answer page in the system

The quality of the question answering system can also be evaluated by pressing the *Evaluate* button for each query individually. The user is provided with the option to mark each answer as $Relevant$ or $Not-relevant$ as shown in Figure 9. The responses are stored in the system as JSON files. The evaluation is discussed further in Section 4

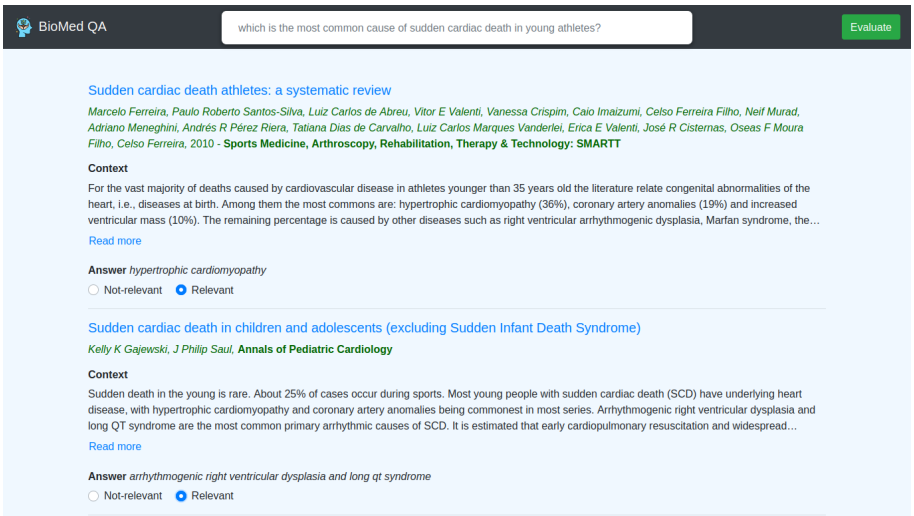

Figure 9: Evaluation page of the system

4 System evaluation

In this section, an explanation of how our model has been evaluated will be provided. First, it was necessary to create a set of relevant questions to evaluate the model. Different types of questions were used to evaluate how robust our model is and whether all question types could be accepted by the model. The types of questions used are factoid, list, yes/no, why,

and how. It should be mentioned, that when creating these questions some topics had to be discarded, e.g. covid-19, as they are not yet included in the database provided by PubMed [4]. The content of the questions was obtained from pathology, cell biology and anatomy exams from the biomedical engineering degree, as well as general medical knowledge questions. In this way we can evaluate our system with more technical as well as simpler questions and see if in both cases the performance is similar. The final set of questions can be observed below:

Factoid questions

- What is a melanoma?
- What is the ideal vascular access for hemodialysis because it presents fewer complications?

List questions

- What are the stages of chronic HIV infection?
- What are the most commonly mutated proto-oncogenes?

Yes/no questions

- Is it possible to run with a torn anterior cruciate ligament?
- Is it possible to live with one kidney?

How questions

- How does cancer spread?
- How does obesity affect my health?

Why questions

- Why is smoking bad?
- Why are X-rays dangerous?

Once the set of questions was created, it was necessary to find experts in the field in order to evaluate the model. For this purpose, 3 biomedical engineers were asked to asses our model. First, before starting the evaluation, they were asked to read the questions carefully, and if they did not understand any of them, they should inform us. In all three cases, all questions were understood without any problems. For all questions, they were expected to know the answer beforehand, or with the context provided along with the answer, be able to discern whether the answer provided was relevant or not. For each of the questions, five answers were provided. These were evaluated as relevant or not relevant as it can be observed in Figure 9.

The results obtained from this evaluation can be found in section 5. It should be mentioned that after analyzing them, it was decided to expand the number of factoid questions. In addition to being the best performed by the system, when we asked the participants what they would like the system to solve for them, most of the questions were of this type. Therefore, 8 more questions were added to the factoid group and their results will also be discussed.

Factoid questions

- What is bcl2?
- What is heparin?
- What is the treatment for asthma?
- Which is the most known bacterium responsible for botulism?
- What is the treatment of acute myocarditis?
- What is the most frequent clinical manifestation of Cytomegalovirus (CMV) infection in AIDS patients?
- What is the main cause of a stroke?
- What is the average age to suffer heart failure?

For evaluation of the system, we select the evaluation metric Precision at rank k ($P@k$) for $k = 1$ to 5. For precision at rank k, we divide the total number of passages with relevant answers that occur at k or below with k. Mathematically,

$$P@k = \frac{c(r)}{k} \tag{1}$$

where $c(r)$ is the count of relevant passages among the retrieved passages up to rank k. We compute the $P@k$ for individual questions as well as provide the average over all the questions. We provide the results in Section 5.

5 Results and discussion

This section presents the results obtained after performing the system evaluation. In addition, a discussion of them will be provided.

Table 2 shows the P@K values for each of the questions presented in section 4. In this table it can be observed that the model's performance varies depending on the questions' category. *Factoid* questions as well as *explanatory* ones —how questions— have a good performance, with at 5 answers being retrieved an at least 3 out of the 5 being relevant. With respect to the *list*, *yes/no* and *why* questions, results can be underwhelming sometimes, with scenarios in which no good answer is retrieved and others in which there is at least one good answer,

but the system is unable to return five candidates. Overall, *list*, *yes/no* and *why* questions have low P@K values —even reaching 0—, whereas *explanatory* and *factoid* have higher ones. As it can be observed in Table 3, these discrepancies in performance, when dealing with the categories, are reflected in the average precision, which is smaller than the *factoid*/*how* ones but greater than the other categories ones.

Category	**Questions**	**P@1**	**P@2**	**P@3**	**P@4**	**P@5**
Factoid	*What is a melanoma?*	1.0	0.5	0.333	0.25	0.4
	What is the ideal vascular access for hemodialysis because it presents fewer complications?	1.0	1.0	0.666	0.75	0.8
How	*How does cancer spread?*	1.0	1.0	1.0	0.75	0.8
	How does obesity affect my health?	1.0	1.0	1.0	1.0	1.0
List	*What are the stages of chronic HIV infection?*	1.0	0.5			
	What are the most commonly mutated proto-oncogenes?	0.0	0.0			
Yes/No	*Is it possible to run with a torn anterior cruciate ligament?*	0.0	0.0	0.0	0.0	0.25
	Is it possible to live with one kidney?	0.0	0.0	0.0	0.0	0.2
Why	*Why is smoking bad?*	0.0	0.0	0.0		
	Why are X-rays dangerous?	0.0	1.0	0.333		

Table 2: Top 5 P@k values for each category question.

P@k	Average Precision
P@1	0.5
P@2	0.5
P@3	0.4165
P@4	0.4583
P@5	0.575

Table 3: Average Precision for the system's top 5 answers in each of the 5 categories questions.

When handling *list* questions, the model is able to return a list as an answer as long as it explicitly contained in the context. If this is not the case, a single response will be provided and, although it is true that this may be relevant, it could be considered incomplete. Something

similar happens with *yes/no* questions. Here, in some cases, an appropriate explanation to the question might be obtained. However, in no case is the answer is transformed into a yes or no statement. This clearly shows that the QA module lacks the ability of more advanced reasoning, and just itself limits to find a more basic suitable response. Finally, for *why* questions, which are also more complex and involve deeper interpretation, the answer needs to be overtly in the text. In other words, if the context does not include the reason, the model is unable to elaborate a sophisticated explanation.

Regarding that no five responses are returned at all times, there are two explanations. The first one is suitable for questions related to younger medical terms or facts. Since the database has limited number of articles, which correspond to documents published longer ago, the document retriever will be unable to find context in which the answer is found, because there are no articles related to that topic. The other case happens when context is found, but the QA model is unable to find an answer in it. Since the QA model is implemented to return information only when an answer is found — and keep looking until there are 5 relevant ones— if there are not answers it will not show any. Again, this problem is probably caused by the lack of deeper interpretation.

As already discussed in section 4, since the factoid questions were the ones that most interested our evaluators, we decided to evaluate a total of 10 factoid questions and see how our system worked. The results obtained can be seen in Table 4. In this case for all questions 5 answers are obtained. Moreover, the average precision increases significantly in comparison as observed in5.

Factoid Questions	**P@1**	**P@2**	**P@3**	**P@4**	**P@5**
What is a melanoma?	1.0	0.5	0.333	0.25	0.4
What is the ideal vascular access for hemodialysis because it presents fewer complications?	1.0	1.0	0.666	0.75	0.8
What is bcl2?	0.0	0.5	0.333	0.5	0.4
What is heparin?	1.0	0.5	0.333	0.5	0.6
What is the treatment for asthma?	1.0	0.5	0.666	0.75	0.6
Which is the most known bacterium responsible for botulism?	1.0	1.0	1.0	1.0	1.0
What is the treatment of acute myocarditis?	0.0	0.5	0.666	0.5	0.6
What is the most frequent clinical manifestation of Cytomegalovirus (CMV) infection in AIDS patients?	1.0	1.0	1.0	1.0	1.0
What is the main cause of a stroke?	0.0	0.5	0.666	0.75	0.8
What is the average age to suffer heart failure?	1.0	1.0	1.0	1.0	0.8

Table 4: Top 5 P@K values for factoid questions.

P@k	Average Precision
P@1	0.7
P@2	0.7
P@3	0.699
P@4	0.7
P@5	0.7

Table 5: Average Precision for the system's top 5 answers in factoid questions

However, although for factoid questions the performance improves significantly, there are still some problems.First, in some answers, although this answer was correct, the context provided casts doubt on this answer or refutes it. An example of this can be seen in Fig10, where although it is true that the answer to this question is *IVGG*, in the text provided it says *"Use of IVGG for acute myocarditis should not be part of routine practice"*.

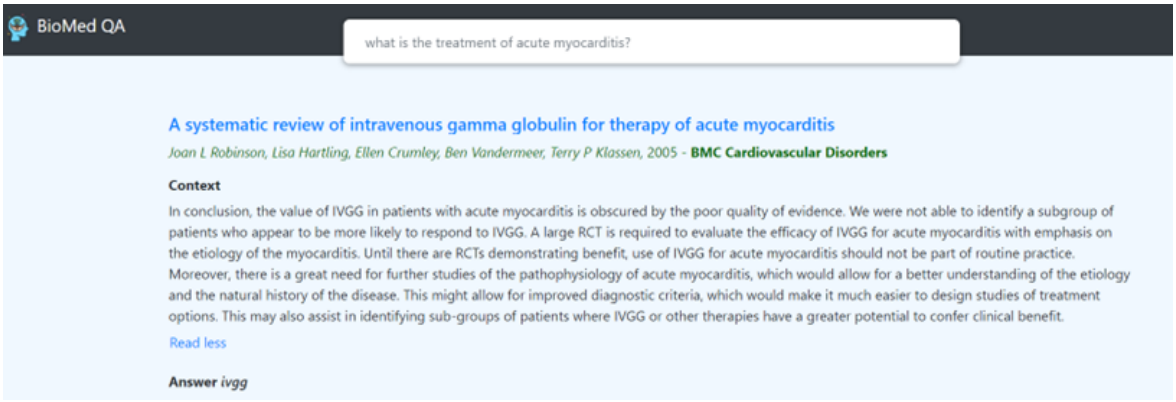

Figure 10: Example where the context does not agree with the correct answer it provides.

Moreover, recent topics have not been included in the database, so questions related to covid-19 cannot be answered. Another example is that due to the recent increase of monkeypox cases, new papers are being published, however, in this case although it is able to find some correct answers or papers, they date from 2003 to 2006 and do not correspond to the new situation.

LAS RESPUESTAS ENTRE ELLAS SE COMPLEMENTAN

DECIR QUE A VECES AUNQUE SI ES CIERTO QUE LA RESPUESTA ES LA CORRECTA, EN EL CONTEXTO SE PONE EN DUDA QUE ESA SEA LA BUENA O INTENTAN REBATIR LO CONTRARIO, SIENDO POR TANTO ESE PAPER NO RELEVANTE

6 Conclusion

It can be thus concluded that the QA model fails to provide a suitable answer when the tasks are more NLP intensive and require an increased level of language, semantics and sintactics reasoning.

References

[1] E. Mutabazi, J. Ni, G. Tang, and W. Cao, "A review on medical textual question answering systems based on deep learning approaches," *Applied Sciences*, vol. 11, no. 12, p. 5456, 2021.

[2] Q. Jin, Z. Yuan, G. Xiong, Q. Yu, H. Ying, C. Tan, M. Chen, S. Huang, X. Liu, and S. Yu, "Biomedical question answering: A survey of approaches and challenges," *ACM Computing Surveys (CSUR)*, vol. 55, no. 2, pp. 1–36, 2022.

[3] D. Su, Y. Xu, T. Yu, F. B. Siddique, E. J. Barezi, and P. Fung, "Caire-covid: A question answering and query-focused multi-document summarization system for covid-19 scholarly information management," *arXiv preprint arXiv:2005.03975*, 2020.

[4] J. Beck and E. Sequeira, "Pubmed central (pmc): An archive for literature from life sciences journals," *The NCBI Handbook*, 2003.

[5] Ncbi.nlm.nih.gov, "FTP Service - PubMed Central," 2022.

[6] S. Yu, Z. Yuan, J. Xia, S. Luo, H. Ying, S. Zeng, J. Ren, H. Yuan, Z. Zhao, Y. Lin, *et al.*, "Bios: An algorithmically generated biomedical knowledge graph," *arXiv preprint arXiv:2203.09975*, 2022.

[7] D. Ostrovsky, M. Haji, and Y. Rodenski, *ElasticSearch Integration*, pp. 159–178. Berkeley, CA: Apress, 2015.

[8] M. Honnibal and I. Montani, "spaCy 3: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing." 2021.

[9] M. Neumann, D. King, I. Beltagy, and W. Ammar, "ScispaCy: Fast and Robust Models for Biomedical Natural Language Processing," in *Proceedings of the 18th BioNLP Workshop and Shared Task*, (Florence, Italy), pp. 319–327, Association for Computational Linguistics, Aug. 2019.

[10] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 11 2019.

[11] K. Ishwari, A. Aneeze, S. Sudheesan, H. Karunaratne, A. Nugaliyadde, and Y. Mallawarrachchi, "Advances in natural language question answering: A review," *arXiv preprint arXiv:1904.05276*, 2019.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[13] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, "Biobert: a pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, 2020.

[14] Y. Gu, R. Tinn, H. Cheng, M. Lucas, N. Usuyama, X. Liu, T. Naumann, J. Gao, and H. Poon, "Domain-specific language model pretraining for biomedical natural language processing," *ACM Transactions on Computing for Healthcare (HEALTH)*, vol. 3, no. 1, pp. 1–23, 2021.

[15] P. Rajpurkar, R. Jia, and P. Liang, "Know what you don't know: Unanswerable questions for squad," *arXiv preprint arXiv:1806.03822*, 2018.

[16] G. Tsatsaronis, G. Balikas, P. Malakasiotis, I. Partalas, M. Zschunke, M. R. Alvers, D. Weissenborn, A. Krithara, S. Petridis, D. Polychronopoulos, *et al.*, "An overview of the bioasq large-scale biomedical semantic indexing and question answering competition," *BMC bioinformatics*, vol. 16, no. 1, pp. 1–28, 2015.