MiBi at BioASQ 2024: Retrieval-Augmented Generation for Answering Biomedical Questions

Notebook for the BioASQ Lab at CLEF 2024

Jan Heinrich Merker[1,†], Alexander Bondarenko[1,2,†], Matthias Hagen[1] and Adrian Viehweger[3]

[1]Friedrich-Schiller-Universität Jena, Germany
[2]Leipzig University, Germany
[3]Institute of Medical Microbiology and Virology, University Hospital Leipzig, Germany

Abstract

In this paper, we describe the MiBi team's participation in the BioASQ 2024 Task 12b on biomedical semantic question answering. Our RAG-based systems (retrieval-augmented generation) use GPT-3.5, GPT-4, or Mixtral to generate an answer from some retrieved context. For the retrieval, we use PubMed's search API or a local BM25 index of PubMed abstracts and potentially re-rank the initially retrieved abstracts / snippets with different neural bi-encoder and cross-encoder re-rankers. We test five different retrieval-augmented generation schemes with different orders of generation and retrieval stages.

The evaluation results for our submitted systems—although partially inconsistent over different test batches—show three general trends. First, combining BM25-based lexical retrieval with neural re-rankers seems to be a good retrieval setup. Second, answers generated from retrieved snippets as context seem more accurate than answers generated from complete abstracts. Third, GPT-4 generates more accurate answers than GPT-3.5, but Mixtral is on par with GPT-4 when employing a generation-then-retrieve-then-generation scheme.

Keywords

Retrieval-augmented generation, biomedical question answering, information retrieval, large language models

1. Introduction

Due to the access to large language models (LLMs) becoming easier, the way people search for answers to all kinds of questions has been changing. Now, it might seem easier to ask ChatGPT and get an instant answer than to use a web search engine and read through the returned search results. While newer and larger LLMs are becoming more accurate, they still can confabulate incorrect facts in their responses. One of the ways to reduce confabulation is a retrieval-augmented generation (RAG) approach [1].

To investigate the RAG effectiveness in the scenario of biomedical question answering, our team MiBi participated in all the phases of the BioASQ 2024 Task 12b on biomedical semantic question answering [2, 3, 4, 5] that includes an abstract retrieval task, a snippet extraction task, and an answer generation task. Particularly for answer generation, we use LLMs (GPT-3.5 [6], GPT-4 [7], and Mixtral-8x7B [8]) and provide them with the context information that contains either retrieved top-3 complete PubMed abstracts or top-10 snippets extracted from the abstracts. The latter approach aims to condense potentially useful, relevant facts and reduce possible context "noise". For retrieving abstracts, we experiment with PubMed's search API and Elasticsearch. Methodically, besides the more conventional retrieval-then-generation paradigm, we also test the generation-then-retrieval paradigm that expands the query before retrieval and re-ranking and more advanced RAG paradigms. Furthermore, for snippet extraction, we either use a chain-of-thought few-shot LLM prompting [9] or heuristically re-rank the PubMed article's titles and chunks of up to three consecutive sentences from the abstracts.[1]

CLEF 2024: Conference and Labs of the Evaluation Forum, September 9–12, 2024, Grenoble, France

†These authors contributed equally.
heinrich.merker@uni-jena.de (J. H. Merker); alexander.bondarenko@uni-jena.de (A. Bondarenko); matthias.hagen@uni-jena.de (M. Hagen); adrian.viehweger@medizin.uni-leipzig.de (A. Viehweger)
0000-0003-1992-8696 (J. H. Merker); 0000-0002-1678-0094 (A. Bondarenko); 0000-0002-9733-2890 (M. Hagen); 0000-0002-8970-5204 (A. Viehweger)

[1]Code, data, and prompts available at https://github.com/webis-de/clef24-mibi-bioasq/

2. Approach

Our BioASQ submissions follow the retrieval-augmented generation paradigm [1], where existing state-of-the-art large language models are given context based on retrieved documents and/or snippets to improve the reliability and factual accuracy of the generated answers. Following the BioASQ's evaluation scheme, we structure our retrieval and generation systems as a modular pipeline consisting of (1) document retrieval and re-ranking, (2) snippet extraction and re-ranking, and (3) answer generation. We explore several implementation options for the three pipeline stages, ranging from simpler heuristic or statistical approaches to complex neural approaches. We also experiment with different paradigms of retrieval-augmented generation, ranging from standard retrieval-then-generate pipelines to more complex RAG pipelines intertwining retrieval and generation.

2.1. Document Retrieval

The first subtask of the BioASQ Task 12b is to retrieve relevant abstracts from the PubMed corpus,[2] an open collection of 37 million medical abstracts.

PubMed API and Neural Re-Ranking Our first document retrieval approach queries the PubMed's search API[3] to retrieve up to 200 abstracts (term matching to achieve higher recall) for further re-ranking (to increase precision at top ranks) using the lower-cased test questions as queries (after removing stop words from NLTK [10] and punctuation). If no document is found for the query, it is iteratively shortened by one word at a time until at least one abstract is found for the shortened query. The resulting abstract(s) are then re-ranked with BM25 [11] (k_1=1.5, b=0.75, concatenated title and abstract, tokenizer and stop words from NLTK). Afterward, the top-50 abstracts are re-ranked again with a MiniLM [12] cross-encoder model[4] trained on the MS MARCO Passage Ranking task [13], and finally the top-25 are again re-ranked with a MPNet [14] bi-encoder model.[5]

Elasticsearch and PyTerrier To avoid issues with the PubMed API (which often returns no results), we index the full PubMed annual baseline[6] of 2024 in Elasticsearch, including the abstracts' metadata (e.g., publication type and MeSH terms[7]). Abstracts (and metadata) are then retrieved by (1) matching the question to the article's title and abstract (Elasticsearch's BM25, k_1=1.2, b=0.75, Unicode tokenization[8]) and (2) matching the medical entities extracted from the question (using ScispaCy [15]) to the abstract's MeSH terms annotated by NLM (same Elasticsearch's BM25). The two queries are combined with an OR operator. Articles without an abstract (or with an empty abstract) are not considered for retrieval. Further, we disallow abstracts from 27 manually selected non-peer-reviewed publication types. The Elasticsearch retrieval using the combined query and filtering is implemented as a PyTerrier [16] module.

2.2. Snippet Extraction and Re-Ranking

After document retrieval, the second subtask is to extract short passages (snippets) from the retrieved abstracts containing a direct answer to the given question or key information required to answer it.

LLM-based Snippet Extraction Our first strategy is to extract snippets with GPT-3.5-turbo (temperature: 0, max tokens: 1000, OpenAI API[9]) chain-of-thought few-shot prompting (the prompt contains

[2]https://pubmed.gov/
[3]https://ncbi.nlm.nih.gov/books/NBK25500/
[4]https://huggingface.co/cross-encoder/msmarco-MiniLM-L6-en-de-v1
[5]https://huggingface.co/sentence-transformers/all-mpnet-base-v2
[6]https://pubmed.gov/download/
[7]https://nlm.nih.gov/mesh/
[8]https://unicode.org/reports/tr29/
[9]https://openai.com/api/

3 examples; refer to Appendix A.3 for the LLM instructions and prompt template). We do not do further re-ranking and simply rely on the ordering returned by the LLM.

Rule-based Snippet Extraction and Neural Re-Ranking As an alternative approach, we develop a rule-based snippet extraction. The document is split into sentences (using NLTK [10]); then, the full title and all combinations of up to 3 consecutive sentences from the abstract are used as candidate snippets. The snippets inherit the retrieval score of the abstract from which they were extracted. Subsequently, we re-rank the top-100 with a TAS-B bi-encoder,[10] then re-rank the top-5 with a duoT5 cross-encoder[11] (we use the PyTerrier implementations of the cross- and bi-encoders).

2.3. Answer Generation

In 2024, the BioASQ phases A+ and B feature two answer generation subtasks evaluated in two settings. Answers should either be given in an 'exact' format based on the question type (e.g., just "yes" or "no" for a yes-no question) or in an 'ideal' format as a short explanatory summary. For both answer types, answers can be generated based on either our self-retrieved/-extracted abstracts and snippets (phase A+) or the ground-truth abstracts and snippets provided by the task organizers (phase B).

Few-Shot Prompting with Function Calling Our first approach for answer generation facilitates function calling with the Instructor library[12] to generate structured LLM responses with GPT-3.5-turbo and GPT-4 (for both temperature: 0, max tokens: 1750, accessed via the OpenAI API). Different zero-shot prompting instructions are used for each answer type (i.e., exact or ideal) and question type (i.e., yes-no, list, factoid, or summary). As the context for generating the answer, we include the concatenated top-3 abstracts or all (top-10) snippets in the prompt. When using the ground-truth abstracts and snippets from the task organizers, the cross-encoder- and bi-encoder-based re-ranking from Section 2.1 is employed before selecting the top-3 abstracts.

Modular LLM Programming with DSPy The second answer generation approach is based on the DSPy LLM programming framework [17]. With DSPy, prompts for the LLM are not manually constructed; instead, the expected signature of inputs and outputs are given as Python classes. Our prediction signature defines the input as the question text, a list of snippets, and (only when generating the 'ideal' answer) the previously generated 'exact' answer. The output signature again varies depending on the answer type (exact, ideal) and question type (yes-no, list, factoid, summary). Based on the declared signature, we generate the answer with DSPy's typed predictions (using the Mixtral-8x7B-Instruct-v0.1 model [8] accessed via the Blablador API[13]). We do not use DSPy's built-in prompt optimization.

2.4. Changing the Order of Retrieval and Answer Generation

So far, our approaches have been based on the typical retrieval-augmented generation paradigm: first, retrieve relevant documents (or snippets) and then generate an answer using the retrieved documents as context. This retrieval-then-generation (RtG) paradigm [18] intuitively assumes that context is often needed to answer complex questions such as medical questions. We challenge this assumption and argue that other paradigms are also valid. For instance, Gienapp et al. [18] also consider the generation-then-retrieval (GtR) paradigm, where an answer is generated first and subsequently grounded by retrieval. Our modular retrieval and answer generation pipeline allows for the evaluation of both RAG patterns.

Additionally, we explore two new variations: retrieval-then-generation-then-retrieval (RtGtR) and generation-then-retrieval-then-generation (GtRtG) that resemble feedback loops [19]. For example, the generated answer can be used to refine the initial retrieval results, similar to how a researcher would

[10] https://huggingface.co/sebastian-hofstaetter/distilbert-dot-tas_b-b256-msmarco
[11] https://huggingface.co/castorini/duot5-base-msmarco
[12] https://useinstructor.com/
[13] https://helmholtz-blablador.fz-juelich.de/

Table 1
Overview of the LLM, retrieval, and answer generation used in our systems for each batch and phase. Full system names are given in Table 3. Parts of the systems implemented but not used in a phase are greyed out. CoT: chain-of-thought; FS: few-shot; 0S: zero-shot; FC: function calling; GT: ground truth; TP: typed prediction. Our most effective configurations per subtask across batches are in bold (cf. Table 2 for results): In phase A, `mibi_a` batch 2 for abstract retrieval, `mibi_4` batch 3 for snippet extraction; in phase A+, `mibi_s` batch 3 for "exact", `mibi_a` batch 3 for "ideal" answers; in phase B, `mibi_s` batch 3 for "exact", `mibi_4` batch 3 for "ideal" answers.

Phase	**System**	**LLM**		**Retrieval**		**Answer**		**Order**
		Model	Tool	Abstracts	Snippets	Gener.	Context	
Test batch 1								
A	mibi_s	GPT-3.5	Instruct.	API, MiniLM, MPNet	$\text{LLM}_{\text{CoT,FS,FC}}$	$\text{LLM}_{\text{0S,FC}}$	snippets	RᴛG
	mibi_a	GPT-3.5	Instruct.	API, MiniLM, MPNet	$\text{LLM}_{\text{CoT,FS,FC}}$	$\text{LLM}_{\text{0S,FC}}$	abs.	RᴛG
A+	mibi_s	GPT-3.5	Instruct.	API, MiniLM, MPNet	$\text{LLM}_{\text{CoT,FS,FC}}$	$\text{LLM}_{\text{0S,FC}}$	snippets	RᴛG
	mibi_a	GPT-3.5	Instruct.	API, MiniLM, MPNet	$\text{LLM}_{\text{CoT,FS,FC}}$	$\text{LLM}_{\text{0S,FC}}$	abs.	RᴛG
B	mibi_s	GPT-3.5	Instruct.	GT, MiniLM, MPNet	GT	$\text{LLM}_{\text{0S,FC}}$	snippets	RᴛG
	mibi_a	GPT-3.5	Instruct.	GT, MiniLM, MPNet	GT	$\text{LLM}_{\text{0S,FC}}$	abs.	RᴛG
Test batch 2								
A	mibi_s	GPT-3.5	Instruct.	Elasticsearch	Rule, TAS-B, duoT5	$\text{LLM}_{\text{0S,TP}}$	snippets	RᴛG
	mibi_a	GPT-3.5	Instruct.	Elasticsearch	Rule, TAS-B, duoT5	$\text{LLM}_{\text{0S,TP}}$	abs.	RᴛG
A+	mibi_s	GPT-3.5	Instruct.	Elasticsearch	Rule, TAS-B, duoT5	$\text{LLM}_{\text{0S,TP}}$	snippets	RᴛG
	mibi_a	GPT-3.5	Instruct.	Elasticsearch	Rule, TAS-B, duoT5	$\text{LLM}_{\text{0S,TP}}$	abs.	RᴛG
B	mibi_s	GPT-3.5	Instruct.	GT, MiniLM, MPNet	GT	$\text{LLM}_{\text{0S,FC}}$	snippets	RᴛG
	mibi_a	GPT-3.5	Instruct.	GT, MiniLM, MPNet	GT	$\text{LLM}_{\text{0S,FC}}$	abs.	RᴛG
Test batch 3								
A	mibi_s	Mixtral	DSPy	Elasticsearch	Rule, TAS-B, duoT5	$\text{LLM}_{\text{0S,TP}}$	snippets	RᴛG
	mibi_a	Mixtral	DSPy	Elasticsearch	Rule, TAS-B, duoT5	$\text{LLM}_{\text{0S,TP}}$	snippets	GᴛR
	mibi_3	Mixtral	DSPy	Elasticsearch	Rule, TAS-B, duoT5	$\text{LLM}_{\text{0S,TP}}$	snippets	RᴛGᴛR
	mibi_4	Mixtral	DSPy	Elasticsearch	Rule, TAS-B, duoT5	$\text{LLM}_{\text{0S,TP}}$	snippets	GᴛRᴛG
	mibi_5	Mixtral	DSPy	Elasticsearch	Rule, TAS-B, duoT5	$\text{LLM}_{\text{0S,TP}}$	snippets	$\text{LLM}_{\text{0S,TP}}$
A+	**mibi_s**	Mixtral	DSPy	Elasticsearch	Rule, TAS-B, duoT5	$\text{LLM}_{\text{0S,TP}}$	snippets	RᴛG
	mibi_a	Mixtral	DSPy	Elasticsearch	Rule, TAS-B, duoT5	$\text{LLM}_{\text{0S,TP}}$	snippets	GᴛR
	mibi_3	Mixtral	DSPy	Elasticsearch	Rule, TAS-B, duoT5	$\text{LLM}_{\text{0S,TP}}$	snippets	RᴛGᴛR
	mibi_4	Mixtral	DSPy	Elasticsearch	Rule, TAS-B, duoT5	$\text{LLM}_{\text{0S,TP}}$	snippets	GᴛRᴛG
	mibi_5	Mixtral	DSPy	Elasticsearch	Rule, TAS-B, duoT5	$\text{LLM}_{\text{0S,TP}}$	snippets	$\text{LLM}_{\text{0S,TP}}$
B	**mibi_s**	GPT-4	Instruct.	GT, MiniLM, MPNet	GT	$\text{LLM}_{\text{0S,FC}}$	snippets	RᴛG
	mibi_a	GPT-4	Instruct.	GT, MiniLM, MPNet	GT	$\text{LLM}_{\text{0S,FC}}$	abs.	RᴛG
	mibi_3	Mixtral	DSPy	GT, Elasticsearch	GT, Rule, TAS-B, duoT5	$\text{LLM}_{\text{0S,TP}}$	snippets	RᴛG
	mibi_4	Mixtral	DSPy	GT, Elasticsearch	GT, Rule, TAS-B, duoT5	$\text{LLM}_{\text{0S,TP}}$	snippets	GᴛRᴛG
	mibi_5	Mixtral	DSPy	GT, Elasticsearch	GT, Rule, TAS-B, duoT5	$\text{LLM}_{\text{0S,TP}}$	snippets	RᴛGᴛR
Test batch 4								
A	mibi_s	Mixtral	DSPy	Elasticsearch	Rule, TAS-B, duoT5	$\text{LLM}_{\text{0S,TP}}$	snippets	RᴛG
	mibi_a	Mixtral	DSPy	Elasticsearch	Rule, TAS-B, duoT5	$\text{LLM}_{\text{0S,TP}}$	snippets	GᴛR
	mibi_3	Mixtral	DSPy	Elasticsearch	Rule, TAS-B, duoT5	$\text{LLM}_{\text{0S,TP}}$	snippets	RᴛGᴛR
	mibi_4	Mixtral	DSPy	Elasticsearch	Rule, TAS-B, duoT5	$\text{LLM}_{\text{0S,TP}}$	snippets	GᴛRᴛG
	mibi_5	Mixtral	DSPy	Elasticsearch	Rule, TAS-B, duoT5	$\text{LLM}_{\text{0S,TP}}$	snippets	$\text{LLM}_{\text{0S,TP}}$
A+	mibi_s	Mixtral	DSPy	Elasticsearch	Rule, TAS-B, duoT5	$\text{LLM}_{\text{0S,TP}}$	snippets	RᴛG
	mibi_a	Mixtral	DSPy	Elasticsearch	Rule, TAS-B, duoT5	$\text{LLM}_{\text{0S,TP}}$	snippets	GᴛR
	mibi_3	Mixtral	DSPy	Elasticsearch	Rule, TAS-B, duoT5	$\text{LLM}_{\text{0S,TP}}$	snippets	RᴛGᴛR
	mibi_4	Mixtral	DSPy	Elasticsearch	Rule, TAS-B, duoT5	$\text{LLM}_{\text{0S,TP}}$	snippets	GᴛRᴛG
	mibi_5	Mixtral	DSPy	Elasticsearch	Rule, TAS-B, duoT5	$\text{LLM}_{\text{0S,TP}}$	snippets	$\text{LLM}_{\text{0S,TP}}$
B	mibi_s	Mixtral	DSPy	GT, Elasticsearch	GT, Rule, TAS-B, duoT5	$\text{LLM}_{\text{0S,TP}}$	snippets	RᴛG
	mibi_a	Mixtral	DSPy	GT, Elasticsearch	GT, Rule, TAS-B, duoT5	$\text{LLM}_{\text{0S,TP}}$	snippets	GᴛR
	mibi_3	Mixtral	DSPy	GT, Elasticsearch	GT, Rule, TAS-B, duoT5	$\text{LLM}_{\text{0S,TP}}$	snippets	RᴛGᴛR
	mibi_4	Mixtral	DSPy	GT, Elasticsearch	GT, Rule, TAS-B, duoT5	$\text{LLM}_{\text{0S,TP}}$	snippets	GᴛRᴛG
	mibi_5	Mixtral	DSPy	GT, Elasticsearch	GT, Rule, TAS-B, duoT5	$\text{LLM}_{\text{0S,TP}}$	snippets	$\text{LLM}_{\text{0S,TP}}$

refine their references based on new findings. In the latter, an answer generated without context is refined after retrieving supporting documents. This resembles roughly a literature review, where an initial hypothesis is checked and refined based on relevant literature.

To facilitate the knowledge aggregation in our approach, we use previously retrieved or generated abstracts, snippets, or answers as follows: For retrieval, the 'exact' and 'ideal' answers are appended to the query. If snippets are given, we use the snippets' origin abstracts for abstract re-ranking (using the same Elasticsearch document scoring as in Section 2.1). Also, previous 'exact' or 'ideal' answers are added to the context according to the DSPy prompt signature (Section 2.3).

Finally, we delegate the decision of which 'task' to execute next to the LLM, using DSPy's typed prediction (Mixtral-8x7B, Blablador API). The LLM is given the question, question type (e.g., yes-no), a history of previously executed "modules", and a readiness flag indicating whether the question is fully answered. Based on this information, the model decides whether to (1) retrieve or re-rank documents, (2) extract and re-rank snippets, (3) generate an 'exact' answer, or (4) generate an 'ideal' answer. Custom DSPy suggestions to the LLM [20] are used to enforce that each module is executed at least once and that the same module cannot execute consecutively (e.g., a retrieval step cannot directly follow another retrieval step to avoid infinite loops). No prompt optimization is used.

2.5. Submitted Systems

We combine the techniques described above in our submitted systems to evaluate the impact of different choices of LLMs and prompting frameworks on the effectiveness of answering biomedical questions. Further systems are submitted to investigate the LLM's reliance on effectively retrieved documents and snippets. Finally, we explore the effect of different retrieval-augmented generation paradigms, with or without feedback loops. An overview of all submitted systems is given in Table 1.

3. Results

We report the evaluation results of our submitted systems in Table 2 using the values for a subset of official BioASQ evaluation metrics published on the task website[14] (recall and MAP for the retrieval phase, macro F_1, strict accuracy, mean precision, and ROUGE-2 F_1 [21] for answer generation). For completeness, we include the 'best', 'worst', and median of the task participants' results. While our results often lie around the median values, two times, we achieved the overall highest results: 'exact' answers for yes-no and factoid questions in phase B of the batch 3, and three times, we scored the worst results: 'exact' answers for factoid questions in phases A+ and B of the batch 2 and 'exact' answers for factoid questions in phase B of the batch 4. In the following, we compare some interesting differences between groups of our systems with respect to the evaluation metrics.

3.1. PubMed API vs. Elasticsearch for Abstract Retrieval

Two different approaches are employed for abstract retrieval: the PubMed search API and our own Elasticsearch index. Comparing our phase A systems `mibi_s` and `mibi_a` from test batch 1 and 2,[15] Table 2 shows a substantial difference in the achieved recall and a noticeable difference in MAP for abstract retrieval. The API-based retrieval used in test batch 1 (cf. Table 1) struggles to find relevant abstracts and consequentially yields a recall below median. We notice that for many queries, the official search API returns too few or no documents at all. Seemingly, it cannot handle the question-like queries well. Although the documents are re-ranked with both cross-encoders and bi-encoders, the MAP scores are only marginally above the median of all submissions. The metadata-enhanced Elasticsearch retrieval, on the other hand, yields above-median recall and MAP even without any re-ranking stage. However, more detailed ablation studies are required to determine what part of the Elasticsearch-based retrieval contributes most to the increase in retrieval effectiveness.

[14] http://bioasq.org/
[15] The same retrieval was used in `mibi_s` and `mibi_a`.

Table 2
Effectiveness of our retrieval and answer generation approaches according to the shared task's official leaderboard. We report recall (Rec.) and mean average precision (MAP) for the retrieval phase (Phase A). For the answer generation phases (Phases A+ and B), we report macro-averaged F_1 of yes-no answers (Ma. F_1), strict accuracy of factual answers (St. A.), mean precision of list answers (M. P.), and ROUGE-2 F_1 of "ideal" answers (R-2 F_1). The "best", "worst", and median of all submissions are given in grey; above-median results are marked in bold. Full system names are given in Table 3. Note that we used different systems in different phases (see Table 1).

System	Phase A				Phase A+				Phase B			
	Abstracts		Snippets		Exact Answer			Ideal	Exact Answer			Ideal
	Rec.	MAP	Rec.	MAP	Yes-no Ma. F_1	Fact. St. A.	List M. P.	Sum. R-2 F_1	Yes-no Ma. F_1	Fact. St. A.	List M. P.	Sum. R-2 F_1
Test batch 1												
best	**0.337**	**0.201**	**0.165**	**0.106**	**0.917**	**0.238**	**0.525**	**0.235**	**0.959**	**0.429**	**0.665**	**0.326**
`mibi_s`	0.156	**0.105**	0.027	**0.041**	**0.750**	0.048	0.218	**0.145**	**0.877**	–	**0.519**	**0.209**
`mibi_a`	0.156	**0.105**	0.027	**0.041**	0.703	0.048	0.271	**0.159**	0.795	–	0.505	0.176
worst	0.034	0.030	0.002	0.000	0.359	0.048	0.173	0.066	0.039	0.048	0.114	0.033
median	0.160	0.107	0.053	0.041	0.737	0.095	0.327	0.114	0.795	0.238	0.505	0.197
Test batch 2												
best	**0.367**	**0.229**	**0.201**	**0.154**	**0.960**	**0.368**	**0.447**	**0.190**	**0.960**	**0.684**	**0.617**	**0.401**
`mibi_s`	**0.258**	**0.158**	**0.095**	**0.051**	0.756	0.158	**0.331**	**0.120**	**0.915**	0.158	**0.544**	**0.244**
`mibi_a`	**0.258**	**0.158**	**0.095**	**0.051**	0.675	0.105	**0.277**	**0.110**	0.876	0.105	0.369	0.179
worst	0.006	0.006	0.000	0.002	0.278	**0.105**	0.077	0.017	0.278	0.105	0.156	0.023
median	0.210	0.115	0.074	0.036	0.756	0.211	0.234	0.105	0.880	0.316	0.500	0.179
Test batch 3												
best	**0.385**	**0.255**	**0.257**	**0.222**	**0.914**	**0.308**	**0.375**	**0.243**	**1.000**	**0.500**	**0.647**	**0.381**
`mibi_s`	**0.235**	**0.139**	0.086	0.052	**0.869**	0.154	**0.256**	**0.190**	**1.000**	0.231	**0.498**	**0.306**
`mibi_a`	**0.228**	**0.125**	**0.101**	**0.055**	0.434	**0.308**	**0.267**	**0.201**	**0.956**	0.231	**0.533**	**0.269**
`mibi_3`	**0.235**	**0.142**	**0.114**	0.046	**0.869**	0.154	**0.256**	**0.186**	0.869	0.154	0.260	0.199
`mibi_4`	**0.228**	0.125	**0.099**	**0.059**	0.534	**0.269**	**0.301**	**0.196**	**0.914**	**0.500**	**0.602**	**0.315**
`mibi_5`	**0.235**	**0.139**	**0.088**	0.046	**0.829**	0.154	**0.249**	**0.189**	0.869	0.154	0.249	0.215
worst	0.004	0.001	0.000	0.014	0.294	0.077	0.102	0.075	0.294	0.039	0.074	0.025
median	0.219	0.125	0.087	0.054	0.765	0.154	0.221	0.160	0.875	0.269	0.421	0.247
Test batch 4												
best	**0.557**	**0.393**	**0.416**	0.344	**0.871**	**0.368**	**0.314**	**0.246**	**0.957**	**0.632**	**0.768**	**0.389**
`mibi_s`	0.230	0.138	0.099	0.045	0.432	0.105	0.148	**0.182**	0.432	0.105	0.148	0.185
`mibi_a`	0.228	0.134	0.095	0.037	0.338	**0.211**	**0.232**	**0.172**	**0.871**	**0.474**	**0.629**	**0.298**
`mibi_3`	0.230	**0.140**	0.100	0.041	0.432	0.105	0.148	**0.181**	0.432	0.105	0.148	0.185
`mibi_4`	0.228	0.134	0.095	0.037	0.338	0.158	**0.239**	**0.163**	**0.871**	**0.579**	**0.667**	**0.277**
`mibi_5`	0.230	0.138	0.100	0.047	0.475	0.105	0.148	**0.167**	0.475	0.105	0.148	0.198
worst	0.018	0.004	0.016	0.003	0.229	0.053	0.081	0.065	0.229	0.105	0.118	0.024
median	0.269	0.138	0.130	0.055	0.682	0.158	0.148	0.123	0.797	0.421	0.454	0.226

3.2. GPT-3.5 vs. Rules and Re-Ranking for Snippet Extraction

Further, we change the snippet extraction approach from test batch 1 to test batch 2. In batch 1, snippets are extracted with GPT-3.5 chain-of-thought few-shot-prompting, while in batch 2, we extract consecutive sentences from each article's title and abstract (chunks) and then re-rank the chunks with a bi-encoder and cross-encoder. In Table 2, the systems using GPT-3.5 for snippet extraction (`mibi_s` and `mibi_a` from batch 1) struggle, especially with respect to the recall of snippets (phase A). Our rule- and re-ranking-based snippet extraction (cf. `mibi_s` and `mibi_a` from batch 2) achieves a higher recall and MAP. Yet, neither of the approaches yields competitive effectiveness with the 'best' of the task's leaderboard. An important limitation of our comparison of snippet extractions is that the abstract retrieval approach has changed simultaneously. So, the snippet extraction effectiveness could also be influenced by the changed abstract retrieval step on which the snippet extraction relies.

3.3. Snippets vs. Abstracts as Context for Answer Generation

In the first three test batches of Phase B, we submitted systems that use either the top-10 snippets or the top-3 (re-ranked) abstracts as context for the LLM to generate an answer (see Table 1). Due to the large length of abstracts (which can even be structured in multiple sections), we assume that shorter snippets would “confuse” an LLM less than full abstracts. Indeed, we can observe that this assumption holds for all submitted systems. The effectiveness benefit of using snippets instead of abstracts is most pronounced in the generation of yes-no answers and ‘ideal’ long-form answers, while for factoid and list answers, the LLM-generated answers are not consistently better with either abstracts or snippets.

3.4. GPT-3.5 vs. GPT-4 vs. Mixtral-8x7B for Answer Generation

A clear effect is also observed when comparing GPT-3.5 with GPT-4 for answer generation (cf. `mibi_s` and `mibi_a` in batch 1/2 and 3 of phase B respectively). Answers generated with GPT-4 are more accurate than the GPT-3.5 answers for all question types except for list questions. Moreover, GPT-4 works (nearly) perfectly for answering yes-no questions when provided with ground-truth relevant context, yielding the highest macro F_1 score of all submissions to phase B in batch 3. A direct comparison of GPT-4 to Mixtral is possible with the systems `mibi_s` and `mibi_3` of test batch 3. Here, Mixtral is outperformed by GPT-4 with respect to effectiveness on all question types. However, Mixtral achieves comparable effectiveness for answering biomedical questions when following the generation-then-retrieval-then-generation paradigm (compare `mibi_s` to `mibi_4`; see Section 3.5). Because we have used Mixtral with another LLM framework (i.e., DSPy programming instead of Instructor prompting) and prompt design (i.e., automatically generated prompts from DSPy instead of manually designed prompts), the comparability of our GPT-based systems to the Mixtral-based systems is rather limited.

3.5. Retrieval-then-Generation vs. Generation-then-Retrieval RAG Paradigms

As our final evaluation, we compare the impact of different RAG paradigms on the retrieval and answer generation effectiveness. This impact is best observed in phases A and A+ of test batch 3 and 4, where we use the same approaches for the retrieval step and the generation step of our RAG pipeline, but perform retrieval and generation in different orders. We either follow the conventional retrieval-then-generation paradigm (RtG), the generation-then-retrieval paradigm (GtR), the new retrieval-then-generation-then-retrieval paradigm (RtGtR, i.e., re-ranking abstracts and snippets based on the LLM’s answer), the new generation-then-retrieval-then-generation paradigm (GtRtG, i.e., refining the LLM’s answer with documents retrieved based on the LLM’s initial answer), or a completely flexible approach where the LLM itself decides in which order to run the retrieval and generation for a given question ($\mathrm{LLM}_{0S,TP}$). Overall, all the paradigms work equally effectively for abstract and snippet retrieval. With respect to answer generation, in phase A+ (no ground-truth context), the best paradigms for yes-no answers were RtG, GtRtG, and $\mathrm{LLM}_{0S,TP}$ while for the remaining answer types, GtR and GtRtG work best. When ground-truth abstracts and snippets are given (phase B), also for yes-no answers GtR and GtRtG work best. Note that our GtR systems do not use retrieved context for answer generation (i.e., an LLM is simply prompted to directly answer the question). Our evaluation results show that the no-context answer generation is often more accurate for ‘ideal’ answers in phase A+ but is less accurate when ground-truth snippets are used as context in phase B. The effectiveness differences between the two system types are, however, marginal. In conclusion, while directly asking an LLM a biomedical question can result in accurate answers, RAG-based approaches provide the possibility to ground the answers in high-quality sources (e.g., PubMed articles), for instance, by providing citations (cf. GtR). Our implementation of the GtRtG paradigm that then refines LLM-generated answers based on these citations is a first step towards a more conclusive evaluation of grounded RAG paradigms.

3.6. Limited Comparability Across Test Batches

We used the same systems in batch 3 and batch 4 (except `mibi_s` and `mibi_a` for phase B). Hence, with comparable test collections, we would expect similar effectiveness for the same systems when comparing batch 3 to batch 4. Somewhat surprisingly, this is not true for answer generation (phases A+ and B) but only for the retrieval in phase A. Looking at the questions from both batches, we find that questions from batch 4 are longer (avg. words: 12.0, non-stopwords: 7.5) and contain more medical entities (avg. entities: 3.7) on average than the questions from batch 3 (words: 10.7, non-stopwords: 6.6, entities: 3.1). More in-depth analyses are hence required, e.g., to check whether the questions from test batch 4 were indeed more difficult than the questions from batch 3. Consequently, our findings that partially rely on the comparability of the different batches would need to be re-evaluated.

4. Conclusion

In this paper, we described the MiBi team's participation in all the subtasks (phases) of the BioASQ 2024 Task 12b on biomedical semantic question answering. In our submitted systems, we used a retrieval-augmented generation approach to enhance the accuracy of LLM-generated answers by providing relevant contextual information retrieved from PubMed, a large collection of 37 million medical abstracts.

For abstract retrieval, we either used PubMed's search API followed by re-ranking with cross- and bi-encoders or a local Elasticsearch index using BM25 as a retrieval model with further filtering based on metadata. We found that the former retrieval strategy was less effective due to a lower recall of the retrieved document candidates. Additionally, our most effective snippet extraction approach was to split the article titles and abstracts into chunks of up to three sentences and rank them using cross- and bi-encoders. This approach was more accurate than prompting an LLM to extract snippets.

Furthermore, for answer generation, we tested several LLMs by providing them with retrieved context. We found that GPT-4 and Mixtral were on par and were more accurate than GPT-3.5. The highest answer accuracy was achieved using only snippets as context, indicating that removing potentially distracting information from abstracts may be beneficial for RAG-based approaches. Comparing different orderings of the retrieval and generation stages, we found the generation-then-retrieval-then-generation paradigm to be the most effective and that retrieved context given upfront tends to "confuse" LLMs. We also found that RAG and no-RAG approaches for answer generation in the BioASQ task settings were almost equally accurate. However, RAG can provide a possibility to reference sources in practical applications.

An interesting future research direction is to explore the effectiveness of using knowledge graphs to steer the LLM-based answer generation. Last, a more systematic evaluation of using different orderings of the retrieval and generation stages in RAG pipelines is worthwhile.

Acknowledgments

This work has been partially supported by the Stiftung für Innovation in der Hochschullehre under the "freiraum 2022" call (FRFMM-58/2022). We are also thankful to Helmholtz AI Jülich for the free access to their Blablador API (for Mixtral inference).

References

[1] P. S. H. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, D. Kiela, Retrieval-augmented generation for knowledge-intensive NLP tasks, in: Proceedings of NeurIPS 2020, NeurIPS Foundation, 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

[2] A. Nentidis, G. Katsimpras, A. Krithara, S. Lima-López, E. Farré-Maduell, M. Krallinger, N. Loukachevitch, V. Davydova, E. Tutubalina, G. Paliouras, Overview of BioASQ 2024: The

twelfth BioASQ challenge on large-scale biomedical semantic indexing and question answering, in: Proceedings of CLEF 2024, 2024.
[3] A. Nentidis, G. Katsimpras, A. Krithara, G. Paliouras, Overview of BioASQ Tasks 12b and Synergy 12 in CLEF 2024, in: Working Notes of CLEF 2024, 2024.
[4] A. Krithara, A. Nentidis, K. Bougiatiotis, G. Paliouras, BioASQ-QA: A manually curated corpus for biomedical question answering, Scientific Data 10 (2023). doi:10.1038/s41597-023-02068-4.
[5] G. Tsatsaronis, G. Balikas, P. Malakasiotis, I. Partalas, M. Zschunke, M. R. Alvers, D. Weissenborn, A. Krithara, S. Petridis, D. Polychronopoulos, Y. Almirantis, J. Pavlopoulos, N. Baskiotis, P. Gallinari, T. Artières, A. N. Ngomo, N. Heino, É. Gaussier, L. Barrio-Alvers, M. Schroeder, I. Androutsopoulos, G. Paliouras, An overview of the BioASQ large-scale biomedical semantic indexing and question answering competition, BMC Bioinform. 16 (2015) 138:1–138:28. doi:10.1186/S12859-015-0564-6.
[6] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language models are few-shot learners, in: Proceedings of NeurIPS 2020, NeurIPS Foundation, 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.
[7] OpenAI, J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, et al. (281 authors), GPT-4 technical report, arXiv 2303.08774, 2023. doi:10.48550/arXiv.2303.08774.
[8] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. de Las Casas, E. B. Hanna, F. Bressand, G. Lengyel, G. Bour, G. Lample, L. R. Lavaud, L. Saulnier, M. Lachaux, P. Stock, S. Subramanian, S. Yang, S. Antoniak, T. L. Scao, T. Gervet, T. Lavril, T. Wang, T. Lacroix, W. E. Sayed, Mixtral of experts, arXiv 2401.04088, 2024. doi:10.48550/arXiv.2401.04088.
[9] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, D. Zhou, Chain-of-thought prompting elicits reasoning in large language models, in: Proceedings of NeurIPS 2022, NeurIPS Foundation, 2022. URL: http://papers.nips.cc/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract.html.
[10] S. Bird, E. Klein, E. Loper, Natural Language Processing with Python, O'Reilly, 2009. URL: https://www.nltk.org/book/.
[11] S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, M. Gatford, Okapi at TREC-3, in: Proceedings of TREC 1994, volume 500-225 of *NIST Special Publication*, NIST, 1994, pp. 109–126. URL: http://trec.nist.gov/pubs/trec3/papers/city.ps.gz.
[12] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, M. Zhou, MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, in: Proceedings of NeurIPS 2020, NeurIPS Foundation, 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.
[13] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, L. Deng, MS MARCO: A human generated machine reading comprehension dataset, in: Proceedings of CoCo@NIPS 2016, volume 1773, CEUR-WS.org, 2016. URL: https://ceur-ws.org/Vol-1773/CoCoNIPS_2016_paper9.pdf.
[14] K. Song, X. Tan, T. Qin, J. Lu, T. Liu, MPNet: Masked and permuted pre-training for language understanding, in: Proceedings of NeurIPS 2020, NeurIPS Foundation, 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/c3a690be93aa602ee2dc0ccab5b7b67e-Abstract.html.
[15] M. Neumann, D. King, I. Beltagy, W. Ammar, ScispaCy: Fast and robust models for biomedical natural language processing, in: Proceedings of BioNLP@ACL 2019, ACL, 2019, pp. 319–327. doi:10.18653/V1/W19-5034.
[16] C. Macdonald, N. Tonellotto, S. MacAvaney, I. Ounis, PyTerrier: Declarative experimentation in python from BM25 to dense retrieval, in: Proceedings of CIKM 2021, ACM, 2021, pp. 4526–4533. doi:10.1145/3459637.3482013.
[17] O. Khattab, A. Singhvi, P. Maheshwari, Z. Zhang, K. Santhanam, S. Vardhamanan, S. Haq, A. Sharma, T. T. Joshi, H. Moazam, H. Miller, M. Zaharia, C. Potts, DSPy: Compiling declarative language model

calls into self-improving pipelines, arXiv 2310.03714, 2023. doi:10.48550/arXiv.2310.03714.
[18] L. Gienapp, H. Scells, N. Deckers, J. Bevendorff, S. Wang, J. Kiesel, S. Syed, M. Fröbe, G. Zuccon, B. Stein, M. Hagen, M. Potthast, Evaluating generative ad hoc information retrieval, in: Proceedings of SIGIR 2024, ACM, 2024. doi:10.1145/3626772.3657849.
[19] D. Arora, A. Kini, S. R. Chowdhury, N. Natarajan, G. Sinha, A. Sharma, GAR-meets-RAG paradigm for zero-shot information retrieval, arXiv 2310.20158, 2023. doi:10.48550/arXiv.2310.20158.
[20] A. Singhvi, M. Shetty, S. Tan, C. Potts, K. Sen, M. Zaharia, O. Khattab, DSPy assertions: Computational constraints for self-refining language model pipelines, arXiv 2312.13382, 2023. doi:10.48550/arXiv.2312.13382.
[21] C.-Y. Lin, ROUGE: A package for automatic evaluation of summaries, in: Text Summarization Branches Out, ACL, Barcelona, Spain, 2004, pp. 74–81. URL: https://aclanthology.org/W04-1013.

A. Appendix

A.1. System Name Mapping

Table 3 maps the shortened system names used in this paper to the full system names used for the submissions to the task.

Table 3
System name shorthands used in this paper and their corresponding full names.

Short name	Full name
mibi_s	mibi_rag_snippet
mibi_a	mibi_rag_abstract
mibi_3	mibi_rag_3
mibi_4	mibi_rag_4
mibi_5	mibi_rag_5

A.2. Publication Types

The publication types listed below (sorted by number of occurrences) were excluded from first-stage retrieval from PubMed:[16]

Letter, Comment, Editorial, News, Biography, Congress, Video-Audio Media, Interview, Overall, Retraction of Publication, Retracted Publication, Newspaper Article, Bibliography, Legal Case, Directory, Personal Narrative, Address, Randomized Controlled Trial (Veterinary), Autobiography, Dataset, Clinical Trial (Veterinary), Festschrift, Webcast, Observational Study (Veterinary), Dictionary, Periodical Index, Interactive Tutorial.

A.3. Snippet Extraction Prompt

The below instructions and few-shot template were used for extracting snippets using GPT-3.5.[17]

LLM instructions

```
"role": "system",
"content": "You are a world class system to extract relevant sentences from titles
    and abstracts answering questions",
"role": "user",
```

[16]Filtering code available at https://github.com/webis-de/clef24-mibi-bioasq/
[17]Full code and prompts available at https://github.com/webis-de/clef24-mibi-bioasq/

```
"content": "Extract from the title and abstract ONLY sentences of phrases that
    directly answer the question",
"role": "user",
"content": <few-shot prompt>,
```

Few-shot prompt template with three examples

```
Here are 3 examples:

[Title]: Rethinking ramoplanin: the role of substrate binding in inhibition of
    peptidoglycan biosynthesis

[Abstract]: Ramoplanin is a cyclicdepsipeptide antibiotic that inhibits
    peptidoglycan biosynthesis. It was proposed in 1990 to block the MurG step of
    peptidoglycan synthesis by binding to the substrate of MurG, Lipid I. The
    proposed mechanism of MurG inhibition has become widely accepted even though it
    was never directly tested. In this paper, we disprove the accepted mechanism for
     how ramoplanin functions, and we present an alternative mechanism. This work
    has implications for the design of ramoplanin derivatives and may influence how
    other proposed substrate binding antibiotics are studied.

[Question]: Which was the first adeno-associated virus vector gene therapy product
    approved
in the United States?

[Extracted]:

Title sentences: [empty list] (no sentences or phrases that directly answer the
    question)
Abstract sentences: [empty list] (no sentences or phrases that directly answer the
    question)

******************

[Title]: Rethinking ramoplanin: the role of substrate binding in inhibition of
    peptidoglycan biosynthesis

[Abstract]: Ramoplanin is a cyclicdepsipeptide antibiotic that inhibits
    peptidoglycan biosynthesis. It was proposed in 1990 to block the MurG step of
    peptidoglycan synthesis by binding to the substrate of MurG, Lipid I. The
    proposed mechanism of MurG inhibition has become widely accepted even though it
    was never directly tested. In this paper, we disprove the accepted mechanism for
     how ramoplanin functions, and we present an alternative mechanism. This work
    has implications for the design of ramoplanin derivatives and may influence how
    other proposed substrate binding antibiotics are studied.

[Question]: Which antibiotics target peptidoglycan biosynthesis?

[Extracted]:

Title sentences: ["Rethinking ramoplanin: the role of substrate binding in
    inhibition of peptidoglycan biosynthesis."]

Abstract sentences: ["Ramoplanin is a cyclicdepsipeptide antibiotic that inhibits
    peptidoglycan biosynthesis."]

******************

[Title]: Mycobacterium Avium Complex (MAC) Lung Disease in Two Inner City Community
     Hospitals: Recognition, Prevalence, Co-Infection with Mycobacterium
    Tuberculosis (MTB) and Pulmonary Function (PF) Improvements After Treatment.
```

[Abstract]: Over 4 years, we evaluated patients who had positive MAC cultures, MAC infection and coinfection with MTB. Lung disease was related/likely related to MAC in 21 patients (50\%) and not related in 21 (50\%). In patients with MAC-related lung disease, the primary physician did not consider the diagnosis except when that physician was a pulmonologist. Half of those with MAC-related lung disease were smokers, white and US-born. There were 12 immunocompetent patients with MTB and NTM cultures. Eleven were non-white and all were foreign-born. Presentation and clinical course were consistent with MTB. All 8 patients with abnormal PF improved. The prevalence of MAC lung infection in two inner city hospitals was four times higher than that of TB. The indication for treatment of MAC infection should also rely heavily on clinical and radiological evidence when there is only one positive sputum culture. The diagnosis was considered only when the admitting physician was a pulmonologist. Most patients with combined infection were clinically consistent with MTB and responded to anti MTB treatment alone. Treatment with anti-MAC therapy improved PF in those patients whose PF was abnormal to begin with.

[Question]: Is Mycobacterium avium less susceptible to antibiotics than Mycobacterium tuberculosis?

[Extracted]:

Title sentences: [empty list] (no sentences or phrases that directly answer the question)
Abstract sentences: ["The prevalence of MAC lung infection in two inner city hospitals was four times higher than that of TB.", "Most patients with combined infection were clinically consistent with MTB and responded to anti MTB treatment alone."]

****

Here is the data:

[Title]: {title}

[Abstract]: {abstract}

[Question]: {question}

[Extracted]: