Webis at TREC 2024: Biomedical Generative Retrieval, Retrieval-Augmented Generation, and Tip-of-the-Tongue Tracks

Maik Fröbe*
Friedrich-Schiller-Universität Jena
Jena, Germany

Lukas Gienapp*
Leipzig University & ScaDS.AI
Leipzig, Germany

Jan Heinrich Merker*
Friedrich-Schiller-Universität Jena
Jena, Germany

Harrisen Scells*
Universität Kassel
Kassel, Germany

Eric Oliver Schmidt*
Martin-Luther-Universität Halle
Halle, Germany

Matti Wiegmann*
Bauhaus-Universität Weimar
Weimar, Germany

Martin Potthast
Universität Kassel
Kassel, Germany

Matthias Hagen
Friedrich-Schiller-Universität Jena
Jena, Germany

Abstract

In this paper, we describe the Webis Group's participation in the 2024 edition of TREC. We participated in the Biomedical Generative Retrieval track, the Retrieval-Augmented Generation track, and the Tip-of-the-Tongue track. For the biomedical track, we applied different paradigms of retrieval-augmented generation with open- and closed-source LLMs. For the Retrieval-Augmented Generation track, we aimed to contrast manual response submissions with fully-automated responses. For the Tip-of-the-Tongue track, we employed query relaxation as in our last year's submission (i.e., leaving out terms that likely reduce the retrieval effectiveness) that we combine with a new cross-encoder that we trained on an enriched version of the TOMT-KIS dataset.

Keywords

Retrieval-Augmented Generation, Biomedical Retrieval, Tip-of-the-Tongue, Large Language Models in Retrieval Systems

1 Introduction

Generative retrieval using large language models (LLMs) might progressively change how people search the Web. From this transformation, retrieval-augmented generation (RAG) emerged as a promising way to combine the attributability of Web search with the ability of large language models to generate concise answers [16]. Most importantly, RAG is seen as a possibility to reduce undesired effects such as hallucinated, factually incorrect answers, or at least trace back by looking at the referenced search results. We investigate the effectiveness of RAG-based generative retrieval systems on three distinct downstream tasks: web search, biomedical search, and tip-of-the-tongue search. Our approaches use various LLMs (GPT-4 [26], GPT-4o [27], and Mistral [14]) and context retrieved with multi-stage retrieval pipelines that employ ChatNoir [4] or Elasticsearch as first stage followed by cascading re-rankers.

For the Biomedical Generative Retrieval track, we submitted seven runs that either apply retrieval-augmentated generation [16], generation-augmented retrieval, or both, using smaller, open-source and larger, closed-source LLMs. Our retrieval uses an Elasticsearch index of 37M biomedical abstracts from the PubMed, enriched with open-access full texts. Instead of manually engineering prompts, we rely on tuned few-shot prompting using DSPy [15].

For the Retrieval-Augmented Generation track, we submitted a total of 15 runs across the three subtasks (5 runs for the retrieval task, 4 runs for the augmented generation task, and 6 runs for the retrieval augmented generation task). Our main motivation was to contrast a manual run with fully automated runs, for which we did create a manual submission for as many topics as our time budget allowed (manually creating a RAG response often took between 1 and 2 hours per topic, we did create responses for 31 topics in approximately 40 hours of work).

For the Tip-of-the-Tongue track, we submitted 5 runs that focused to improve our submission from last year [6]. Last year, we participated with long-query reduction approaches aiming at the idea to remove terms from the query that confuse the retrieval model, i.e., improving the recall by making the query smaller. Therefore, we improved our TOMT-KIS dataset and combined a cross-encoder trained on our new improved dataset with our best query reduction approach from last year.

2 Biomedical Generative Retrieval Track

In our seven submissions to the TREC Biomedical Generative Retrieval (BioGen) track, we explored different variants of retrieval-augmented generation (RAG) [16] based on PubMed articles and using two different proprietary, and open-source LLMs.[1]

2.1 Approach

Our RAG system employs a retrieval module using PyTerrier [21] and Elasticsearch, that retrieves, ranks, and extracts relevant context passages for a given biomedical question, and a generation module using DSPy [15], that generates an answer to the question based on the retrieved context. For the submitted runs, the retrieval and generation modules were combined in different RAG paradigms, going from retrieve-then-generate and generate-then-retrieve paradigms to a more iterative refinement of the retrieved

*These authors contributed to the paper equally and are listed alphabetically.

The Thirty-Third Text REtrieval Conference, TREC 2024, November 18–22, 2022, Rockville, Maryland
2024.

[1] Code and data available at https://github.com/webis-de/trec24-biogen/

Table 1: Parameter choices or ranges for the hyperparameter optimization of our BioGen runs.

Parameter	Range
Retrieval	
Use topic question in query	yes
Use topic title in query	yes/no
Use topic narrative in query	yes/no
Use summary answer in query	yes/no
Use exact answer in query	yes/no
Remove query stop words	yes/no
Match on PubMed title	must/should
Match on PubMed abstract	must/should
Match on PubMed full text	must/should
Match on PubMed MeSH terms	must/should
Filter non-empty PubMed title	yes/no
Filter non-empty PM abstract	yes/no
Filter peer-reviewed pub. types	yes/no
Extract passages from abstract	yes/no
Extract passages from full text	yes/no
Passage max. sentences	1–3
Pointwise re-ranker	monoT5 [25], TAS-B [12], ANCE [33], TCT-ColBERT [18], none
Pairwise re-ranker	duoT5 [28], none
Generation	
LLM	Mistral-7B [14], GPT-4o mini [27]
Context references cutoff	3, 5, 10
Summary answer prompting	few-shot, CoT few-shot
Exact answer prompting	few-shot, CoT few-shot
Augmentation	
Augmentation type	cross-augmentation, independent augmentation, none
Augmentation steps	1–3
Back-augmentation	yes/no

passages and generated answers in multiple augmentation rounds, building on our previous work [11, 22].

Specifically, our modularized approach allows to change the usual retrieve-then-generate order of doing RAG [11, 16] to more flexible execution orders like generate-then-retrieve [3] or (by iteratively augmenting the retrieval and generation steps), even new RAG paradigms like retrieve-then-generate-then-retrieve, or similar. To this end, we implement two approaches: (1) independent augmentation, where the generation module is (optionally iteratively) augmented by a number of retrieval steps and vice-versa, but also (2) "cross-augmentation", where the output of the first retrieval step is used to augment the second generation step and the output of the first generation step is used to augment the second retrieval step. Both paradigms allow for repeated application, the goal being that the information from both retrieval and generation "converge" towards the user's information need.

All our retrieval runs use an index of 37 million scientific abstracts from the 2024 PubMed baseline,[2] 21 million of which are included in the subset selected by the TREC BioGen organizers. We enriched 3.4 million articles with their full texts by querying the OpenAlex API[3]—the same API used by popular browser extension Unpaywall[4]—for open-access PDF sources of each article, and subsequently extracting the text from the downloaded PDF using pypdf[5]. The final index of 792 GB is hosted on a 130-node Elasticsearch cluster, and contains 2.8 million full texts for the TREC BioGen-selected subset of the PubMed.

We used Optuna [1] and DSPy [15] to jointly tune model choices (e.g., RAG paradigm, retrieval model) and hyperparameters (e.g., number of augmentation rounds, number of few-shot-examples) based on 10 randomly sampled instances from the training data of the BioASQ lab at CLEF 2024 [31] (BioASQ 12b). While the questions from the BioASQ 12b instances could be used as is for the TREC BioGen task, the ground-truth answers from BioASQ do not give references in the text. Yet, at least 10 ground-truth context passages were given per question. Hence, to be able to tune our RAG systems to cite references, for each BioASQ answer, we first constructed a list of references by using the PubMed IDs of all given, de-duplicated context passages. This references list was then added to each sentence from the BioASQ's ground-truth answer, as split by spaCy's sentence parser [23].[6]

To cover both proprietary and open-source LLMs, we performed the hyperparameter tuning separately with the larger GPT-4o mini model [27] and the smaller Mistral-Instruct-v0.3 model [14].[7] For both LLMs, we first ran 100 trials and ranked the trials by the geometric mean of two retrieval-focused measures, Recall@1000 and nDCG [13] (evaluated using `ir_measures` [19]), and two generation-focused measures, ROUGE-1 F1 and ROUGE-L F1 [17]. Then, for each LLM, the top-10 trials are again evaluated for the Recall@1000, nDCG, ROUGE-1 F1, ROUGE-L F1, and two additional LLM-based measures, faithfulness and answer relevance (from the RAGAS toolkit [5]). For submission, we selected up to five of the best, non-equivalent[8] trials per LLM, again ranked by the geometric mean of all six evaluated measures.

For the selected hyperparameter choices, we finally run the system again on all topics from the current TREC BioGen track and store the answers in the specified JSON format. In the JSON output of the selected systems, we then semi-automatically fixed minor format issues in the generated answers by: (1) Removing dangling references, (2) merging wrongly split sentences (e.g., in `implantation and pregnancy. [19464684].`), and (3) removing repeated dots or spaces.

2.2 Submitted Runs

We submitted seven runs, four using the Mistral-Instruct-v0.3 LLM and three using the GPT-4o mini model:

[2] https://pubmed.gov/download/
[3] https://docs.openalex.org/
[4] https://unpaywall.org/
[5] https://pypi.org/project/pypdf/
[6] Model: `en_core_web_sm`
[7] GPT-4o mini accessed via the OpenAI API; Mistral-Instruct-v0.3 accessed via the free Blablador API: https://helmholtz-blablador.fz-juelich.de/
[8] E.g., when certain modules were not used due to the hyperparameter choice.

webis-1. This run retrieves up to ten PubMed articles from our index of 21 M valid abstracts. As the query, we use the concatenated question, title, and narrative from the topic, as well as (when augmenting a previous answer) the simple yes-no, factual, or list answer (if the question type is known). No stop words are removed from the query. The query is matched against just the article's abstract text using Elasticsearch's BM25. We exclude non-peer-reviewed or non-human-health-related publications by a manually curated disallow list.[9] After retrieval, passages are extracted from the retrieved article's abstract text by splitting it into sentences and returning all sentence *n*-grams up to three sentences. The top-50 passages from Elasticsearch are subsequently re-ranked with a (pointwise) TCT-ColBERT model[10] [18].

For generation, the run generates a summary answer for each question with DSPy using a Mistral model[11] [14]. The question and the top-10 passages are given to the model as context (numbered according to their rank after retrieval and re-ranking), and the model is prompted to return a summary answer with (numbered) references given in the text. Using DSPy, we optimize the prompt by labeled few-shot prompting with three examples from the BioASQ 12b training set. After generation, the internal (rank-based) reference numbering is converted back to PubMed IDs according to the format required by TREC BioASQ.

In this run, retrieval and generation are independently augmented with the other, generation and retrieval, respectively. For generation-augmented retrieval, we augment three times while not feeding back retrieval results to the generation module. For retrieval-augmented generation, we also augment three times, but feed back generation results to the retrieval module.

webis-2. Our second run uses the same retrieval as in *webis-1* except for not using the topic title as the query and applying an additional (pairwise) re-ranking step using a duoT5 model[12] [28]. Similarly, the generation step is the same as in *webis-1* except for only giving the model the top-3 passages and using chain-of-thought prompting [32]. Again, the generation prompts are optimized with DSPy. The run uses the same augmentation setup as *webis-1*.

webis-3. This run similarly retrieves up to ten articles from Elasticsearch, and uses the concatenated question and narrative from the topic, as well as previous exact answer (yes/no, factual, or list) as the query. No stop words are removed from the query. In this run, the query is matched against the article's abstract text and the article title (both title and abstract must match). Additionally, the MeSH terms[13] extracted from the query (using scispaCy's medical entity recognition [24]) are matched to the MeSH terms of the indexed PubMed abstracts ("should" match). Like in the previous runs, non-peer-reviewed or non-human-health-related publications are excluded. For passage extraction, we again use sentence *n*-grams of up to three sentences plus the article's title as another passage. The top-10 passages are re-ranked using monoT5[14] [25]. The generation and RAG approaches are the same as for run *webis-1*.

webis-5. Run *webis-5* uses a similar retrieval as *webis-3* except for only matching the query to the abstract and title (title should match, abstract must match; no matching of MeSH terms). Before passage splitting, also articles with an empty (or without) title are excluded. The generation approach and RAG paradigm are the same as for runs *webis-1* and *webis-3*.

webis-gpt-1. This generation-only run generates answers with GPT-4o [27] and does not use any retrieval ("vanilla" generation). Just the topic's question is prompted to the LLM. We optimize the prompt by labeled few-shot prompting with three examples from the BioASQ 12b train set, using DSPy. As no retrieval is used, also no RAG paradigm applies.

webis-gpt-4. In contrast to *webis-gpt-1*, here we again retrieve up to ten articles from Elasticsearch, using the concatenated question, narrative, and previous exact answer as the query. This time, stop words are removed from the query (using spaCy's stop word list [23]). The query is matched only against the article's abstract text and articles with an empty abstract are excluded. No filtering is applied based on publication types. For passage extraction, we again use sentence *n*-grams of up to three sentences. The top-50 passages are re-ranked with a monoT5 model[15] [25].

We give the top-3 re-ranked passages to the model as additional context for answering the question, again using GPT-4o for generation. The unoptimized, templated prompt from DSPy is used without few-shot examples. The retrieval and generation of this run are jointly cross-augmented in two rounds, meaning that twice the generated answer is used as additional query input and the retrieved passages are used as additional context for generation.

webis-gpt-6. The retrieval of this run is similar to *webis-gpt-4*, but uses only the topic's question and previous answer as the query, does not remove stop words, and further excludes non-peer-reviewed or non-human-health-related publications. Moreover, only the top-10 passages are re-ranked by monoT5, and the top-3 of monoT5 is again re-ranked with a duoT5 model [28].[16] Generation for this run is the same as *webis-gpt-5* except that DSPy is used to tune a labeled few-shot-prompting with one example. This run uses independent augmentation, doing generation-augmented retrieval in two rounds (feeding back retrieved results to the generation model) and retrieval-augmented generation in a single round.

2.3 Results

Submissions to the BioGen track were evaluated based on the answer accuracy, quality, and completeness, and based on citation quality and document relevance. Table 2 shows the number and fraction of acceptable answers, answer precision, redundancy, harmfulness, and completeness in terms of recall on three scenarios: (1) strict recall, counting only required and supported sentences, (2) lenient recall, that counts all required sentences, and (3) relaxed

[9] Disallowed publication types: 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.

[10] https://hf.co/castorini/tct_colbert-v2-hnp-msmarco

[11] Model version: `Mistral-7B-Instruct-v0.3`; via Blablador API.

[12] https://hf.co/castorini/duot5-base-msmarco

[13] https://nlm.nih.gov/mesh/

[14] https://hf.co/castorini/monot5-base-msmarco

[15] https://hf.co/castorini/monot5-base-msmarco

[16] https://hf.co/castorini/duot5-base-msmarco

Table 2: Answer generation and retrieval effectiveness results of our runs submitted to the BioGen track, compared to the track's best, mean, and worst results per measure. Measured are the number of acceptable answers (# OK), answer accuracy (Acc.), precision (Prec.), redundancy (Red.), harmfulness (Harm.), and completeness / recall (strict: only required and supported sentences; lenient: all required sents.; relaxed: required and borderline sents.; using either Sentence Transformer (ST) or SimCSE (SCSE) embeddings). For citations, we report coverage (Cov.), and support/contradict rates (SR/CR); for referenced documents recall (Rec.) and precision (Prec.). Better than average results are highlighted in bold.

System	Answer Acc.		Answer Quality			Answer Completeness / Recall						Citation Quality			Doc. Rel.	
	# OK	Acc.	Prec.	Red.	Harm.	Strict		Lenient		Relaxed		Cov.	SR	CR	Rec.	Prec.
						ST	SCSE	ST	SCSE	ST	SCSE					
webis-1	53	0.82	0.66	0.12	**0.00**	0.07	0.08	0.10	0.10	0.11	0.11	0.53	0.42	0.04	0.03	0.51
webis-2	43	0.66	0.53	0.12	**0.00**	0.05	0.05	0.08	0.08	0.08	0.08	0.34	0.42	0.05	0.01	0.52
webis-3	39	0.60	0.53	**0.07**	**0.00**	0.06	0.06	0.08	0.08	0.08	0.08	0.43	0.45	0.05	0.01	0.47
webis-5	43	0.66	0.56	**0.09**	**0.00**	0.03	0.03	0.08	0.08	0.09	0.09	0.29	0.35	**0.01**	0.01	0.41
webis-gpt-4	54	0.83	0.64	0.17	**0.00**	0.07	0.07	0.14	0.14	0.15	0.14	0.42	**0.69**	**0.02**	0.02	**0.78**
webis-gpt-6	60	0.92	0.63	0.26	**0.00**	0.07	0.07	0.16	0.16	0.17	0.17	0.32	0.38	0.04	0.03	0.55
webis-gpt-1	**65**	**1.00**	**0.82**	0.15	**0.00**	0.00	0.00	0.20	0.20	0.20	0.21	0.00	0.00	**0.00**	0.00	0.00
Best	65	1.00	0.91	0.04	0.00	0.43	0.43	0.46	0.47	0.46	0.47	0.92	0.80	0.00	0.24	0.90
Mean	60	0.93	0.77	0.11	0.00	0.20	0.21	0.27	0.27	0.28	0.28	0.64	0.57	0.02	0.08	0.66
Worst	39	0.60	0.53	0.26	0.02	0.00	0.00	0.08	0.08	0.08	0.08	0.00	0.00	0.06	0.00	0.00

recall, where borderline sentences are considered in addition to required sentences. The three answer recall scenarios are each evaluated using either Sentence Transformer[17] [29, 30] or SimCSE[18] [10] embeddings. Citations are evaluated based on their coverage and support or contradict rates, respectively. For retrieval, recall and precision of documents referenced in the answer were considered.

Across many evaluated metrics, our Mistral-based runs yield below-average results. Mistral particularly struggles with answer completeness and precision. From the GPT-based runs, the "vanilla" run, *webis-gpt-1*, that did not use any retrieval, outperformed all runs in terms of answer accuracy (perfect accuracy on the 65 evaluated topics) and precision. Overall, our GPT-based runs generated more accurate, precise, and complete answers than our Mistral-based runs. Yet, GPT generates more redundant answers while not substantially improving answer completeness compared to our Mistral runs. Our Mistral runs yield overall less redundant but also less precise answers. All of our runs struggle with recall, both on the answer level (i.e., low answer completeness) and on the referenced document level (i.e., few relevant documents referenced). Looking at our runs, no clear pattern regarding the RAG paradigm used nor the re-ranking of the retrieval-step can be identified.

We note, however, that the document relevance evaluation is biased by the RAG pipeline's generation step, as participants were only allowed to include in their run documents that were also referenced in the generated answer. In future editions of the task, an insightful addition would be an independent evaluation of the retrieval step, decoupled from which documents were actually referenced from generated answer. We believe that a document which is not cited can still influence the answer generation (e.g., the answer might reference a survey article but still also be based on individual studies that were summarized in the survey).

[17] https://hf.co/sentence-transformers/all-mpnet-base-v2
[18] https://hf.co/princeton-nlp/sup-simcse-roberta-large

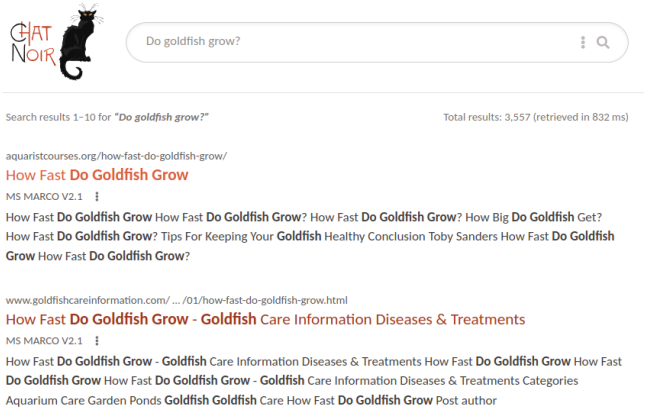

Figure 1: Overview of the ChatNoir search engine result pages that we used for TREC RAG.

3 Retrieval-Augmented Generation Track

We use ChatNoir [4] into which we indexed MS MARCO v2.1 (see Figure 1). Documents in ChatNoir indexed differently than in Anserini [34] following the draft pull request to ir_datasets [20] that indexes each segment via its default_text attribute that is identical to the implementation of MS MARCO v2. We re-rank with monoT5 [28] dockerized from TIRA / TIREx [7, 9].

3.1 Submissions to the Retrieval Task

webis-01. We use multiple systems to create a re-ranking pool for MonoT5 and MonoElectra that are subsequently fused and re-ranked with RankZephyr. The re-ranking pool was created by fusing the results of traditional retrieval systems with a learned dense

model and automatically created boolean query variants retrieved against traditional retrieval systems and additionally enriched by corpus graph retrieval. For the traditional retrieval, we submitted the original queries against Anserini (BM25, INL2, QLD) and ChatNoir (BM25F with a boost for Wikipedia). For the dense retrieval, we used weaviate. We created boolean query variants by using GPT-4o-mini and Llama3.1 by first extracting potential aspects of the query and subsequently generating boolean queries with the LLMs to capture those aspects, the boolean queries were retrieved against ChatNoir. We did re-rank the pools with monoT5-3b and MonoElectra, and used the top-results for-adaptive re-ranking against ChatNoir (i.e., the corpus graph concept). The top-100 monoT5 and monoElectra documents were re-ranked with RankZephyr yielding two runs that we fused with reciprocal rank fusion. On this run, we again re-ranked the top-100 results with RankZephyr, using cascading re-ranking (i.e., re-rank the results of RankZephyr multiple times, we stopped after three iterations). For retrieval, we used the segment, headings, and titles as text. For re-ranking (i.e., with MonoT5, MonoElectra, and RankZephyr), we used only the segment text, i.e., not the title and headings.

webis-02. This run aims to increase the recall base, therefore, the run only consists of documents that are not retrieved within the top-1000 of BM25, QLD, INL2 as implemented in Anserini, BM25F as implemented in ChatNoir, and the top-1000 of our weaviate implementation (dense retrieval). The documents were retrieved via adaptive re-ranking (i.e., the corpus graph) of the top results of RankZephyr and our boolean query formulation (as used in the run *webis-01*). To not waste judgment budget, we only include documents that make it into the top-75 of our *webis-01* run (that incorporated cascading re-ranking). For some topics that did not retrieve new documents we pad with the baseline.

webis-03. This is our run *webis-01* but diversified so that each segment is removed for which a neighbouring segment was already retrieved. This aims to ensure that an LLM (for the retrieval augmented generation) sees more diverse retrieval content.

webis-04. This is our run *webis-01* but diversified so that per page only the top-segment retrieved. This aims to ensure that an LLM (for the retrieval augmented generation) sees more diverse retrieval content.

webis-05. We use multiple systems to create a re-ranking pool for MonoElectra. The re-ranking pool was created by fusing the results of traditional retrieval systems with a learned dense model and automatically created boolean query variants retrieved against traditional retrieval systems and additionally enriched by corpus graph retrieval. For the traditional retrieval, we submitted the original queries against Anserini (BM25, INL2, QLD) and ChatNoir (BM25F with a boost for Wikipedia). For the dense retrieval, we used weaviate. We created boolean query variants by using GPT-4o-mini and Llama3.1 by first extracting potential aspects of the query and subsequently generating boolean queries with the LLMs to capture those aspects, the boolean queries were retrieved against ChatNoir. We did re-rank the pools with MonoElectra.

3.2 Submissions to the Augmented Generation Task

webis-taskrag-gpt4omini-k10. We decompose the RAG pipeline into 3 individual generation tasks: (1) 'Extract' yields the most salient information form a doc given a query-doc pair; (2) 'Combine' merges the extracted information of two docs; and (3) 'Condense' reformulates the merged evidence into a final response. The pipeline first applies extract to each document, then combines all documents with pairwise merges in a tree-like fashion. Prompts for each task are shown in Table 3. Documents are merged in relevance ordering, i.e., first taking each subsequent pair (rank 1 and rank 2, rank 3 and rank 4, ...), and then recursively applying that to the newly combined texts. On the overall resulting text, i.e., the root node, we apply the condense task to infer the final response. Attribution is achieved via prompting the model to include explicit references, i.e., [0], at each step. References are then parsed using regex to conform with the final submission format. Prompts were formulated using an iterative manual reformulation approach, with feedback regarding the quality of each prompted task at each step. For this run, we use the top-10 ranked documents, and iteratively prompt OpenAI's GPT-4o-mini model.

webis-taskrag-gpt4omini-k20. This run uses the same approach as the previous one, but instead using the top-20 documents.

webis-reuserag-promptedreuse-clustered. This run uses the baseline retrieval run as retrieval input. For generation, we split all sentences into 3 groups based on the prompt sentences by calculating semantic similarity with SBERT. We then concatenate the top ranked sentences together to form the response. Baseline introduction, middle, and conclusions sentences were used as 'prompts' to cluster sentences into 3 groups.

webis-reuserag-baseline-promptedreuse-clustered. Segments from the baseline run were clustered automatically using SBERT embeddings. The top ranked sentences from each cluster were concatenated to form the response.

3.3 Submissions to the Retrieval Augmented Generation Task

webis-manual. We did create manual responses for 31 topics (ca. 40 hours of manual work; creating a manual response for a topic often takes between 1 and 2 hours per topic). The responses that are padded from the baseline are from baseline_rag24.test_gpt-4o_top20 without any modification.

webis-taskrag-zephyr-gpt4omini-k10. This run follows the same approach to generation as describe for the run `webis-taskrag-gpt4omini-k10` submitted to the augmented generation task. Yet, instead, it uses the top-10 documents of the `webis-01` retrieval run.

webis-taskrag-zephyr-gpt4omini-k20. This run uses the same approach as the previous one, but instead using the top-20 documents.

webis-taskrag-zephyr-gpt4omini-k10-shuffled. This run uses the *webis-01* retrieval run as retrieval input, and additionally shuffles the set of retrieved documents. For generation, we decompose the RAG pipeline into 3 individual generation tasks. 'Extract' yields the most salient information form a doc given a query-doc pair;

Table 3: Task prompts as used in our TaskRAG approach.

Task	Prompt
Extract	`You will be provided with two texts: a query and a context. Your task is to generate a text that reflects the parts of the context that best answers the query, possibly rephrasing or shortening it. The context may contain IDs noted in brackets that indicate sources, for example [1]; keep these IDs intact and correctly reference sources. The answer must include the context's given ID. Do not insert facts, knowledge, or information not contained in the context. Reply only with the answer. Only generate full sentences. If no information relevant to query is found, reply with '<empty>'`
Combine	`You will be provided with two texts: a first context and a second context. Your task is to generate a text that combines both contexts into a single coherent text, shortening or summarizing key points if necessary. The order of contexts can be changed if necessary. The contexts may contain IDs noted in brackets that indicate sources, for example [1] or [1,2]; keep these IDs intact and correctly reference sources. Each sentence should be attributed with relevant sources. The answer must include a context's given IDs if used. Do not insert facts, knowledge, or information not contained inthe context. Reply only with the answer. Only generate full sentences.`
Condense	`You will be provided with two texts: a query, and a context. Your task is to generate a single coherent conclusive answer to the query based on the information given in the context. Include as much information from the context as possible. The context may contain IDs noted in brackets that indicate sources, for example [1] or [1,2]; keep these IDs intact and correctly reference sources. Each sentence should be attributed with relevant sources. Do not insert facts, knowledge, or information not contained in the context. Reply only with the answer. Only generate full sentences.`

'Combine' merges the extracted information of two docs; 'Condense' reformulates the merged evidence into a final response. The pipeline first applies extract to each document, then combines all documents with pairwise merges in a tree-like fashion, and finally condense the final response. Attribution is achieved via prompting the model to include explicit references, i.e., [0], at each step. References are then parsed using regex to conform with the final submission format. Prompts were formulated using an iterative manual reformulation approach, with feedback regarding the quality of each prompted task at each step.

webis-taskrag-zephyr-llama31-k10. This run uses the *webis-01* retrieval run as retrieval input. For generation, we decompose the RAG pipeline into 3 individual generation tasks. 'Extract' yields the most salient information form a doc given a query-doc pair; 'Combine' merges the extracted information of two docs; 'Condense' reformulates the merged evidence into a final response. The pipeline first applies extract to each document, then combines all documents with pairwise merges in a tree-like fashion, and finally condense the final response. Attribution is achieved via prompting the model to include explicit references, i.e., [0], at each step. References are then parsed using regex to conform with the final submission format. Prompts were formulated using an iterative manual reformulation approach, with feedback regarding the quality of each prompted task at each step.

Table 4: Effectiveness of our 5 runs in the retrieval task of TREC RAG.

Approach	Recall@1000	nDCG@10	RR
webis-01	0.8102	0.6590	0.9006
webis-02	0.1067	0.4277	0.8755
webis-03	0.4825	0.6423	0.9041
webis-04	0.3200	0.6340	0.9049
webis-05	0.8063	0.6370	0.9048

Table 5: Effectiveness of our 5 runs in the augmented generation task of TREC RAG.

Approach	nDCG@10
webis-ag-run0-taskrag	0.5345
webis-ag-run1-taskrag	0.6006
webis-ag-run3-reuserag	0.6330
webis-ag-run2-reuserag	0.6330

Table 6: Effectiveness of our 5 runs in the retrieval augmented generation task of TREC RAG.

Approach	nDCG@10
webis-rag-run3-taskrag	0.4936
webis-rag-run1-taskrag	0.6253
webis-manual	0.5857
webis-rag-run0-taskrag	0.5637
webis-rag-run4-reuserag	0.6590
webis-rag-run5-reuserag	0.6590

webis-reuserag-promptedreuse-k10. This run uses the *webis-01* retrieval run as retrieval input. For generation, split all sentences into 3 groups based on the prompt sentences by calculating semantic similarity with SBERT. We then concatenate the top ranked sentences together to form the response. Baseline introduction, middle, and conclusions sentences were used as 'prompts' to cluster sentences into 3 groups.

webis-reuserag-promptedreuse-clustered. The segments from the *webis-01* run were clustered automatically using SBERT embeddings. The top ranked sentences from each cluster were concatenated to form the response.

3.4 Results

Table 4 shows the results for the retrieval task of TREC RAG. Table 5 shows the results for the augmented generation task. Table 6 shows the results for the retrieval augmented generation task.

4 Tip-of-the-Tongue Track

We submit 5 runs to the Tip-of-the-Tongue track. All our submissions use ChatNoir, a BM25-F search engine that retrieves on the text and the title as first stage system (we re-use parameters tuned on the ClueWeb09 as this is the default of ChatNoir. We use query relaxation implemented with GPT-4o-mini in multiple variants to generate relaxed queries by instructing the large language model to leave out terms that likely reduce the retrieval effectiveness. Our approaches were informed by our submission to the 2023 edition [2], i.e., we only applied query relaxation approaches that worked well in the previous year. All our submitted runs did not use the official baseline runs. For two runs, we derived a triplet dataset from the existing TOMT-KIS dataset [8].[19] The triplet dataset is available on Hugging Face,[20] and suitable for training of cross-encoder models (for every TOMT-KIS query one positive and one negative document). This TOMT-KIS dataset has overlapping queries to the test set removed and uses used BM25 retrieval against the title of wikipedia articles to include positive matches that have no explicit link to Wikipedia. In cases when we used monoT5 as re-ranker, we used the versions dockerized from TIRA / TIREx [7, 9].

4.1 Submitted Approaches

Our 5 runs are:

webis-base. This run uses ChatNoir as first-stage retrieval system. We use the union of six queries submitted against ChatNoir (retrieving always the top-1000 per query), the original query and five long query reduction approaches (i.e., GPT-4o-mini prompted in different ways to reduce the long original query). All retrieved results are subsequently re-ranked by monoT5-base (castorini/monot5-base-msmarco-10k) using the original query.

webis-tot-01. This run uses ChatNoir as first-stage retrieval system. We use the union of six queries submitted against ChatNoir (retrieving always the top-1000 per query), the original query and five long query reduction approaches. All retrieved results are re-ranked by monoT5-base (castorini/monot5-base-msmarco-10k) base using the original query and two reduced queries. For each of those monoT5-base scored-queries, we re-score the top-100 with monoT5-3b against the original query that yield our top-results. We fill up with the monoT5-base scored queries.

webis-tot-02. This run uses the first stage as *webis-base* to score candidates for six queries submitted against ChatNoir (original query and five long query reduction approaches). We re-score all top-100 candidates with monoT5-3b against the original query and two reduced queries, yielding 3 scores per top-document that we subsequently fuse with min-max-normalized reciprocal rank fusion implemented in ranx. We fill up with the monoT5-base scored queries.

[19]https://hf.co/datasets/webis/tip-of-my-tongue-known-item-search
[20]https://hf.co/datasets/webis/tip-of-my-tongue-known-item-search-triplets

Table 7: Effectiveness of our 5 runs in the Tip-of-the-Tongue track.

Approach	Recall@1000	Recall@100	nDCG@10
webis-base	0.6633	0.2750	0.0487
webis-tot-01	0.8200	0.4567	0.0652
webis-tot-02	0.8200	0.7550	0.3193
webis-tot-03	0.8200	0.5083	0.0534
webis-tot-04	0.8200	0.7450	0.4588

webis-tot-03. We use the initial stages as in *webis-tot-01* and re-score the top-100 candidates with a DeBerta model trained on the TOMT-KIS dataset.

webis-tot-04. We use the initial stages as in *webis-tot-01* and re-score all top-100 candidates with monoT5-3b that used the query relaxation strategy that was the most effective in our 2023 submission.

4.2 Results

Table 7 shows the effectiveness in terms of Recall at 1000 and at 100 and nDCG@10 on the Tip-of-the-Tongue track.

References

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of KDD 2019*, Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis (Eds.). ACM, New York, 2623–2631. https://doi.org/10.1145/3292500.3330701

[2] Jaime Arguello, Samarth Bhargav, Fernando Diaz, Evangelos Kanoulas, and Bhaskar Mitra. 2023. Overview of the TREC 2023 Tip-of-the-Tongue Track. In *Proceedings of TREC 2023 (NIST Special Publication)*, Ian Soboroff and Angela Ellis (Eds.). NIST, Gaithersburg, 13 pages. https://trec.nist.gov/pubs/trec32/papers/Overview_tot.pdf

[3] Daman Arora, Anush Kini, Sayak Ray Chowdhury, Nagarajan Natarajan, Gaurav Sinha, and Amit Sharma. 2023. GAR-meets-RAG Paradigm for Zero-Shot Information Retrieval. arXiv 2310.20158. , 18 pages. https://doi.org/10.48550/arXiv.2310.20158

[4] Janek Bevendorff, Benno Stein, Matthias Hagen, and Martin Potthast. 2018. Elastic ChatNoir: Search Engine for the ClueWeb and the Common Crawl. In *Proceedings of ECIR 2018 (Lecture Notes in Computer Science, Vol. 10772)*, Gabriella Pasi, Benjamin Piwowarski, Leif Azzopardi, and Allan Hanbury (Eds.). Springer, Berlin, 820–824. https://doi.org/10.1007/978-3-319-76941-7_83

[5] Shahul ES, Jithin James, Luis Espinosa Anke, and Steven Schockaert. 2024. RAGAs: Automated Evaluation of Retrieval Augmented Generation. In *Proceedings of EACL 2024*, Nikolaos Aletras and Orphée De Clercq (Eds.). ACL, Kerrville, 150–158. https://aclanthology.org/2024.eacl-demo.16

[6] Maik Fröbe, Christine Brychcy, Elisa Kluge, Eric Oliver Schmidt, and Matthias Hagen. 2023. Webis at TREC 2023: Tip-of-the-Tongue Track. In *Proceedings of TREC 2023 (NIST Special Publication)*, Ian Soboroff and Angela Ellis (Eds.). NIST, Gaithersburg, 3 pages. https://trec.nist.gov/pubs/trec32/papers/Webis.T.pdf

[7] Maik Fröbe, Jan Heinrich Reimer, Sean MacAvaney, Niklas Deckers, Simon Reich, Janek Bevendorff, Benno Stein, Matthias Hagen, and Martin Potthast. 2023. The Information Retrieval Experiment Platform. In *Proceedings of SIGIR 2023*. ACM, New York, 2826–2836. https://doi.org/10.1145/3539618.3591888

[8] Maik Fröbe, Eric Oliver Schmidt, and Matthias Hagen. 2023. A Large-Scale Dataset for Known-Item Question Performance Prediction. In *Proceedings of QPP++@ECIR 2023 (CEUR Workshop Proceedings, Vol. 3366)*, Guglielmo Faggioli, Nicola Ferro, Josiane Mothe, and Fiana Raiber (Eds.). CEUR-WS.org, Aachen, 13–19. https://ceur-ws.org/Vol-3366/paper-03.pdf

[9] Maik Fröbe, Matti Wiegmann, Nikolay Kolyada, Bastian Grahm, Theresa Elstner, Frank Loebe, Matthias Hagen, Benno Stein, and Martin Potthast. 2023. Continuous Integration for Reproducible Shared Tasks with TIRA.io. In *Proceedings of ECIR 2023 (LNCS)*, Jaap Kamps, Lorraine Goeuriot, Fabio Crestani, Maria Maistro, Hideo Joho, Brian Davis, Cathal Gurrin, Udo Kruschwitz, and Annalina Caputo (Eds.). Springer, Berlin, 236–241. https://doi.org/10.1007/978-3-031-28241-6_20

[10] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In *Proceedings of EMNLP 2021*, Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (Eds.). ACL, Kerrville, 6894–6910. https://doi.org/10.18653/V1/2021.EMNLP-MAIN.552

[11] Lukas Gienapp, Harrisen Scells, Niklas Deckers, Janek Bevendorff, Shuai Wang, Johannes Kiesel, Shahbaz Syed, Maik Fröbe, Guido Zuccon, Benno Stein, Matthias Hagen, and Martin Potthast. 2024. Evaluating Generative Ad Hoc Information Retrieval. In *Proceedings of SIGIR 2024*. ACM, New York, 14 pages. https://doi.org/10.1145/3626772.3657849

[12] Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. 2021. Efficiently Teaching an Effective Dense Retriever with Balanced Topic Aware Sampling. In *Proceedings of SIGIR 2021*, Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai (Eds.). ACM, New York, 113–122. https://doi.org/10.1145/3404835.3462891

[13] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.* 20, 4 (2002), 422–446. https://doi.org/10.1145/582415.582418

[14] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. arXiv 2310.06825. , 9 pages. https://doi.org/10.48550/arXiv.2310.06825

[15] Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2023. DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines. arXiv 2310.03714. , 32 pages. https://doi.org/10.48550/arXiv.2310.03714

[16] Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Proceedings of NeurIPS 2020*. NeurIPS Foundation, San Diego, 16 pages. https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html

[17] Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*. ACL, Barcelona, Spain, 74–81. https://aclanthology.org/W04-1013

[18] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2021. In-Batch Negatives for Knowledge Distillation with Tightly-Coupled Teachers for Dense Retrieval. In *Proceedings of RepL4NLP@ACL-IJCNLP 2021*, Anna Rogers, Iacer Calixto, Ivan Vulic, Naomi Saphra, Nora Kassner, Oana-Maria Camburu, Trapit Bansal, and Vered Shwartz (Eds.). ACL, Kerrville, 163–173. https://doi.org/10.18653/V1/2021.REPL4NLP-1.17

[19] Sean MacAvaney, Craig Macdonald, and Iadh Ounis. 2022. Streamlining Evaluation with ir-measures. In *Proceedings of ECIR 2022 (LNCS, Vol. 13186)*, Matthias Hagen, Suzan Verberne, Craig Macdonald, Christin Seifert, Krisztian Balog, Kjetil Nørvåg, and Vinay Setty (Eds.). Springer, Berlin, 305–310. https://doi.org/10.1007/978-3-030-99739-7_38

[20] Sean MacAvaney, Andrew Yates, Sergey Feldman, Doug Downey, Arman Cohan, and Nazli Goharian. 2021. Simplified Data Wrangling with ir_datasets. In *Proceedings of SIGIR 2021*, Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai (Eds.). ACM, New York, 2429–2436. https://doi.org/10.1145/3404835.3463254

[21] Craig Macdonald, Nicola Tonellotto, Sean MacAvaney, and Iadh Ounis. 2021. PyTerrier: Declarative Experimentation in Python from BM25 to Dense Retrieval. In *Proceedings of CIKM 2021*. ACM, New York, 4526–4533. https://doi.org/10.1145/3459637.3482013

[22] Jan Heinrich Merker, Alexander Bondarenko, Matthias Hagen, and Adrian Viehweger. 2024. MiBi at BioASQ 2024: Retrieval-Augmented Generation for Answering Biomedical Questions. In *Working Notes of CLEF 2024 (CEUR Workshop Proceedings, Vol. 3740)*, Guglielmo Faggioli, Nicola Ferro, Petra Galuscakova, and Alba García Seco de Herrera (Eds.). CEUR-WS.org, Aachen, 176–187. https://ceur-ws.org/Vol-3740/paper-16.pdf

[23] Ines Montani, Matthew Honnibal, Matthew Honnibal, Adriane Boyd, Sofie Van Landeghem, and Henning Peters. 2023. explosion/spaCy: v3.7.2: Fixes for APIs and requirements. https://doi.org/10.5281/zenodo.10009823

[24] Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. 2019. ScispaCy: Fast and Robust Models for Biomedical Natural Language Processing. In *Proceedings of BioNLP@ACL 2019*. ACL, Kerrville, 319–327. https://doi.org/10.18653/V1/W19-5034

[25] Rodrigo Frassetto Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document Ranking with a Pretrained Sequence-to-Sequence Model. In *Findings of EMNLP 2020*, Trevor Cohn, Yulan He, and Yang Liu (Eds.). ACL, Kerrville, 708–718. https://doi.org/10.18653/v1/2020.findings-emnlp.63

[26] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2023. GPT-4 Technical Report. arXiv 2303.08774. , 100 pages. https://doi.org/10.48550/arXiv.2303.08774

[27] OpenAI, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander M?dry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, Alex Renzin, Alex Tachard Passos, Alexander Kirillov, Alexi Christakis, Alexis Conneau, Ali Kamali, Allan Jabri, Allison Moyer, Allison Tam, Amadou Crookes, Amin Tootoochian, Amin Tootoonchian, Ananya Kumar, Andrea Vallone, Andrej Karpathy, Andrew Braunstein, Andrew Cann, Andrew Codispoti, Andrew Galu, Andrew Kondrich, Andrew Tulloch, Andrey Mishchenko, Angela Baek, Angela Jiang, Antoine Pelisse, Antonia Woodford, Anuj Gosalia, Arka Dhar, Ashley Pantuliano, Avi Nayak, Avital Oliver, Barret Zoph, Behrooz Ghorbani, Ben Leimberger, Ben Rossen, Ben Sokolowsky, Ben Wang, Benjamin Zweig, Beth Hoover, Blake Samic, Bob McGrew, Bobby Spero, Bogo Giertler, Bowen Cheng, Brad Lightcap, Brandon Walkin, Brendan Quinn, Brian Guarraci, Brian Hsu, Bright Kellogg, Brydon Eastman, Camillo Lugaresi, Carroll Wainwright, Cary Bassin, Cary Hudson, Casey Chu, Chad Nelson, Chak Li, Chan Jun Shern, Channing Conger, Charlotte Barette, Chelsea Voss, Chen Ding, Cheng Lu, Chong Zhang, Chris Beaumont, Chris Hallacy, Chris Koch, Christian Gibson, Christina Kim, Christine Choi, Christine McLeavey, Christopher Hesse, Claudia Fischer, Clemens Winter, Coley Czarnecki, Colin Jarvis, Colin Wei, Constantin Koumouzelis, Dane Sherburn, Daniel Kappler, Daniel Levin, Daniel Levy, David Carr, David Farhi, David Mely, David Robinson, David Sasaki, Denny Jin, Dev Valladares, Dimitris Tsipras, Doug Li, Duc Phong Nguyen, Duncan Findlay, Edede Oiwoh, Edmund Wong, Ehsan Asdar, Elizabeth Proehl, Elizabeth Yang, Eric Antonow, Eric Kramer, Eric Peterson, Eric Sigler, Eric Wallace, Eugene Brevdo, Evan Mays, Farzad Khorasani, Felipe Petroski Such, Filippo Raso, Francis Zhang, Fred von Lohmann, Freddie Sulit, Gabriel Goh, Gene

Oden, Geoff Salmon, Giulio Starace, Greg Brockman, Hadi Salman, Haiming Bao, Haitang Hu, Hannah Wong, Haoyu Wang, Heather Schmidt, Heather Whitney, Heewoo Jun, Hendrik Kirchner, Henrique Ponde de Oliveira Pinto, Hongyu Ren, Huiwen Chang, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian O'Connell, Ian Osband, Ian Silber, Ian Sohl, Ibrahim Okuyucu, Ikai Lan, Ilya Kostrikov, Ilya Sutskever, Ingmar Kanitscheider, Ishaan Gulrajani, Jacob Coxon, Jacob Menick, Jakub Pachocki, James Aung, James Betker, James Crooks, James Lennon, Jamie Kiros, Jan Leike, Jane Park, Jason Kwon, Jason Phang, Jason Teplitz, Jason Wei, Jason Wolfe, Jay Chen, Jeff Harris, Jenia Varavva, Jessica Gan Lee, Jessica Shieh, Ji Lin, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joanne Jang, Joaquin Quinonero Candela, Joe Beutler, Joe Landers, Joel Parish, Johannes Heidecke, John Schulman, Jonathan Lachman, Jonathan McKay, Jonathan Uesato, Jonathan Ward, Jong Wook Kim, Joost Huizinga, Jordan Sitkin, Jos Kraaijeveld, Josh Gross, Josh Kaplan, Josh Snyder, Joshua Achiam, Joy Jiao, Joyce Lee, Juntang Zhuang, Justyn Harriman, Kai Fricke, Kai Hayashi, Karan Singhal, Katy Shi, Kavin Karthik, Kayla Wood, Kendra Rimbach, Kenny Hsu, Kenny Nguyen, Keren Gu-Lemberg, Kevin Button, Kevin Liu, Kiel Howe, Krithika Muthukumar, Kyle Luther, Lama Ahmad, Larry Kai, Lauren Itow, Lauren Workman, Leher Pathak, Leo Chen, Li Jing, Lia Guy, Liam Fedus, Liang Zhou, Lien Mamitsuka, Lilian Weng, Lindsay McCallum, Lindsey Held, Long Ouyang, Louis Feuvrier, Lu Zhang, Lukas Kondraciuk, Lukasz Kaiser, Luke Hewitt, Luke Metz, Lyric Doshi, Mada Aflak, Maddie Simens, Madelaine Boyd, Madeleine Thompson, Marat Dukhan, Mark Chen, Mark Gray, Mark Hudnall, Marvin Zhang, Marwan Aljubeh, Mateusz Litwin, Matthew Zeng, Max Johnson, Maya Shetty, Mayank Gupta, Meghan Shah, Mehmet Yatbaz, Meng Jia Yang, Mengchao Zhong, Mia Glaese, Mianna Chen, Michael Janner, Michael Lampe, Michael Petrov, Michael Wu, Michele Wang, Michelle Fradin, Michelle Pokrass, Miguel Castro, Miguel Oom Temudo de Castro, Mikhail Pavlov, Miles Brundage, Miles Wang, Minal Khan, Mira Murati, Mo Bavarian, Molly Lin, Murat Yesildal, Nacho Soto, Natalia Gimelshein, Natalie Cone, Natalie Staudacher, Natalie Summers, Natan LaFontaine, Neil Chowdhury, Nick Ryder, Nick Stathas, Nick Turley, Nik Tezak, Niko Felix, Nithanth Kudige, Nitish Keskar, Noah Deutsch, Noel Bundick, Nora Puckett, Ofir Nachum, Ola Okelola, Oleg Boiko, Oleg Murk, Oliver Jaffe, Olivia Watkins, Olivier Godement, Owen Campbell-Moore, Patrick Chao, Paul McMillan, Pavel Belov, Peng Su, Peter Bak, Peter Bakkum, Peter Deng, Peter Dolan, Peter Hoeschele, Peter Welinder, Phil Tillet, Philip Pronin, Philippe Tillet, Prafulla Dhariwal, Qiming Yuan, Rachel Dias, Rachel Lim, Rahul Arora, Rajan Troll, Randall Lin, Rapha Gontijo Lopes, Raul Puri, Reah Miyara, Reimar Leike, Renaud Gaubert, Reza Zamani, Ricky Wang, Rob Donnelly, Rob Honsby, Rocky Smith, Rohan Sahai, Rohit Ramchandani, Romain Huet, Rory Carmichael, Rowan Zellers, Roy Chen, Ruby Chen, Ruslan Nigmatullin, Ryan Cheu, Saachi Jain, Sam Altman, Sam Schoenholz, Sam Toizer, Samuel Miserendino, Sandhini Agarwal, Sara Culver, Scott Ethersmith, Scott Gray, Sean Grove, Sean Metzger, Shamez Hermani, Shantanu Jain, Shengjia Zhao, Sherwin Wu, Shino Jomoto, Shirong Wu, Shuaiqi, Xia, Sonia Phene, Spencer Papay, Srinivas Narayanan, Steve Coffey, Steve Lee, Stewart Hall, Suchir Balaji, Tal Broda, Tal Stramer, Tao Xu, Tarun Gogineni, Taya Christianson, Ted Sanders, Tejal Patwardhan, Thomas Cunninghman, Thomas Degry, Thomas Dimson, Thomas Raoux, Thomas Shadwell, Tianhao Zheng, Todd Underwood, Todor Markov, Toki Sherbakov, Tom Rubin, Tom Stasi, Tomer Kaftan, Tristan Heywood, Troy Peterson, Tyce Walters, Tyna Eloundou, Valerie Qi, Veit Moeller, Vinnie Monaco, Vishal Kuo, Vlad Fomenko, Wayne Chang, Weiyi Zheng, Wenda Zhou, Wesam Manassra, Will Sheu, Wojciech Zaremba, Yash Patil, Yilei Qian, Yongjik Kim, Youlong Cheng, Yu Zhang, Yuchen He, Yuchen Zhang, Yujia Jin, Yunxing Dai, and Yury Malkov. 2024. GPT-4o System Card. arXiv 2410.21276. , 33 pages. https://doi.org/10.48550/arXiv.2410.21276

[28] Ronak Pradeep, Rodrigo Frassetto Nogueira, and Jimmy Lin. 2021. The Expando-Mono-Duo Design Pattern for Text Ranking with Pretrained Sequence-to-Sequence Models. arXiv 2101.05667. , 23 pages. https://doi.org/10.48550/arXiv.2101.05667

[29] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the EMNLP-IJCNLP 2019*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). ACL, Kerrville, 3980–3990. https://doi.org/10.18653/V1/D19-1410

[30] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. MPNet: Masked and Permuted Pre-training for Language Understanding. In *Proceedings of NeurIPS 2020*. NeurIPS Foundation, San Diego, 11 pages. https://proceedings.neurips.cc/paper/2020/hash/c3a690be93aa602ee2dc0ccab5b7b67e-Abstract.html

[31] George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R. Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, Yannis Almirantis, John Pavlopoulos, Nicolas Baskiotis, Patrick Gallinari, Thierry Artières, Axel-Cyrille Ngonga Ngomo, Norman Heino, Éric Gaussier, Liliana Barrio-Alvers, Michael Schroeder, Ion Androutsopoulos, and Georgios Paliouras. 2015. An overview of the BioASQ large-scale biomedical semantic indexing and question answering competition. *BMC Bioinform.* 16 (2015), 138:1–138:28. https://doi.org/10.1186/S12859-015-0564-6

[32] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Proceedings of NeurIPS 2022*. NeurIPS Foundation, San Diego, 14 pages. http://papers.nips.cc/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract.html

[33] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. In *Proceedings of ICLR 2021*. OpenReview.net, Appleton, 16 pages. https://openreview.net/forum?id=zeFrfgyZln

[34] Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the Use of Lucene for Information Retrieval Research. In *Proceedings of SIGIR 2017*, Noriko Kando, Tetsuya Sakai, Hideo Joho, Hang Li, Arjen P. de Vries, and Ryen W. White (Eds.). ACM, New York, 1253–1256. https://doi.org/10.1145/3077136.3080721