Laboratory for Analytic Sciences in TREC 2025 RAG and RAGTIME Tracks

Yue Wang,[1] John M. Conroy,[2] Neil Molino,[2] Julia Yang,[2] Mike Green[3]

[1]University of North Carolina at Chapel Hill
[2]IDA Center for Computing Sciences
[3]Department of Defense, USA
wangyue@unc.edu, {conroy,npmolin,jsyang}@super.org, magree22@ncsu.edu

Abstract

This report describes submissions by the Laboratory for Analytic Sciences to the TREC 2025 RAG and RAGTime tracks. By leveraging autonomous agent workflows, including query decomposition, planner-executor architectures, and ensemble retrieval techniques (e.g., BM25, SPLADE, T5 sentence embeddings), we examine whether "Agentic RAG" can surpass traditional RAG systems in terms of retrieval relevance, groundedness, factuality, and citation quality. Our evaluations using Open-RAG-Eval, Autonuggetizer, and other metrics indicate gains in nugget coverage, groundedness and citation accuracy, albeit with trade-offs in retrieval relevance and factual consistency. In addition, we explored trade-offs in retrieval methodologies for the TREC RAG retrieval-only task, and agentic generation for the Report Generation task in RAGTIME.

Keywords: Agentic RAG; AI Agents; Retrieval-Augmented Generation; Information Retrieval; LLM Evaluation; TREC; Natural Language Processing; Autonomous Agents; Autogen; Prompt Engineering

1 Introduction

Large language models (LLMs) have been demonstrating strong capabilities in generating fluent natural language answers to complex questions posed by human users. However, LLM-generated answers can be hallucinated, outdated, and lack source attribution [39, 2]. Retrieval-augmented generation (RAG) is a key approach to mitigating these issues. RAG refers to a family of techniques in which an LLM can retrieve data from external sources when responding to questions [10]. RAG mitigates the issues of hallucinated and outdated answers by providing relevant and updated content obtained from quality-controllable and updated data sources (e.g., Web search and enterprise databases) and allows the LLM to attribute its answers to relevant source documents [21, 36]. It has become a predominant approach to connecting people with large amounts of unstructured information, adopted by new systems, including ChatGPT, AI overview in Google search and Bing search, and deep research features offered by OpenAI, Google, and Perplexity, among many other similar services. The rapid adoption of RAG technology calls for evaluation. However, how to properly evaluate the intermediate and final outcomes of RAG technology is still an evolving research frontier.

The National Institute of Standards and Technology (NIST) recognizes the importance of RAG evaluation and started the first RAG track at the 2024 Text Retrieval Conference (TREC) [15, 28, 27, 18]. In 2025, this interest expanded into several tracks. These include a reiteration of last year's RAG track, where the information need and corpus collection are both in English, and a new track called RAGTIME, where the information need is expressed in English, while the corpus collection is multilingual. In both tracks, participating teams build their RAG systems that can generate long-form answers in English to complex questions by summarizing information retrieved from the large corpus collection.

This paper provides an overview of the problem setups, datasets, and evaluation methodologies used in the RAG and RAGTIME tracks. Building off of the team's results from the 2024 competition, we chose to explore an agentic approach to implementing RAG solutions for the two tracks. The paper includes a description of AI agents and our methods to implementing and evaluating agentic systems in order to create TREC submissions.

2 Problem Setup

2.1 Problem Setup for the RAG track

The task in the TREC 2025 RAG track can be formulated as follows. Given a complex question or query q, and a text collection $C = \{x_j\}_{j=1}^{n}$, generate a natural language answer $a = \{(s_i, X_i)|X_i \subset C\}$, where a consists of a sequence of sentences $[s_1, \cdots, s_i, \cdots]$ and each sentence s_i is attributed to (or grounded in) a set of textual segments X_i retrieved from C. The total length of the answer, $|a| = \sum_i |s_i|$, should be no more than l words. The total number of cited textual segments, $|\cup_i X_i|$, should be no more than a fixed budget k.

The RAG track decomposes the above problem into three subtasks [27]:

- **Retrieval task (R)**: given query q, retrieve at most $k = 100$ relevant segments.
- **Augmented generation task (AG)**: given query q and top 100 retrieved segments $X \subset C$ retrieved by a baseline system, summarize these segments into an answer $a = \{(s_i, X_i)|X_i \subset X\}$. Each answer has a length limit $l \leq 400$ words.
- **Retrieval-augmented generation task (RAG)**: the full task pipeline that encompasses both retrieval and augmented generation tasks.

2.1.1 Datasets

RAG track organizers provide the following datasets [30].

Text collection C: RAG track organizers provide MS MARCO V2.1 as the text collection, which was used in the 2024 RAG track. It contains the title, headings, and body of 10,960,555 frequently visited web pages, a subset of the ClueWeb22 collection [17]. Each webpage document is divided into sliding windows of 10-sentence segments with an overlap of 5-sentences [5]. This results in a total of 113,520,750 segments in the entire collection. Although in principle, users can use their own segmentation schemes, TREC organizers ask teams to normalize their segments back to these 10-sentence segments as a common basis for evaluating retrieval and source attribution [27].

Queries $\{q\}$: The development queries ("topics" in TREC parlance) include 301 queries used in the 2024 TREC RAG Track. These are complex questions that need to be answered by possibly synthesizing information from multiple webpages. The track organizers provided segment-level relevance judgments ("qrels" in TREC parlance) for these queries.

In July 2025, RAG track organizers further released example topics to be used this year [31]. Each topic contains a few sentences narrating an information need. An example is the following:

> "I want a thorough understanding of what makes up a community, including its definitions in various contexts like science and what it means to be a 'civilized community.' I'm also interested in related terms like 'grassroots organizations,' how communities set boundaries and priorities, and their roles in important areas such as preparedness and nation-building."

This type of topic is very different from web search queries used in previous years. Web search queries, even those that ask "why" and "how" questions, are much shorter and less nuanced than the narrative above.

2.1.2 Evaluation methodology

The track organizers developed a four-pronged evaluation methodology in the 2024 TREC RAG track [26]: **nugget evaluation**, **support evaluation**, **fluency**, and **retrieval evaluation**. It is likely (but not confirmed) that this evaluation methodology will be used in 2025. The workflow for constructing the gold-standard evaluation data is as follows [26]. For each topic,

1. A pool of segments is constructed by taking the union of retrieved segments found in the answers submitted by different participating teams.

2. The pool of segments are judged automatically (by LLMs) or semi-automatically (by human validation of LLM judgments) [34]. The relevance judgments take a graded scale: *perfectly relevant*, *highly relevant*, *related*, and *irrelevant*. These judged segments become the basis ("qrels") for **retrieval evaluation**.

3. The set of perfectly and highly relevant segments are pooled as an input to an automatic nugget extraction algorithm [19]. This algorithm uses a large language model to examine the pool of relevant segments and extract nuggets, or atomic content units (each in the form of a short sentence), to a cumulative set. At the same time, the algorithm rates each nugget as *vital* or *okay*. The extracted nuggets are then manually curated by human assessors.

4. Given the set of vital and okay nuggets, and an answer submitted by a participating team, a large language model will judge whether the answer (partially) covers the set of nuggets. These become the basis for **nugget evaluation** (precision and recall). The nugget precision and recall are then manually assessed by human assessors.

5. To assess whether each sentence in an answer is well-supported by the cited segment(s), a large language model is used to assess the degree of logical support/entailment. The **support evaluation** is then manually assessed by human assessors.

6. To assess the **fluency** of an answer, a large language model is used by taking into consideration the topic and the answer.

This evaluation plan aligns with recent literature that emphasizes a multi-criteria approach to evaluating large language models [11, 2]. It also provides guidance to developing various components in a RAG system: the retrieval/reranking component should aim to optimize both recall and precision of relevant textual segments provided to the summarizer (segment-level and nugget-level precision and recall); the summarizer or other components need to ensure that each sentence is properly attributed to source segments (support evaluation) and the language should be fluent and within the length limit (fluency evaluation).

2.2 Problem Setup for the RAGTIME track

The task in the TREC 2025 RAGTIME track can be formulated as follows [33]. Given a query (called "report request" in RAGTIME) q expressed in English, and a multilingual text collection $C = \{x_j\}_{j=1}^{n}$, generate a report $a = \{(s_i, X_i) | X_i \subset C\}$, where a consists of a sequence of English sentences $[s_1, \cdots, s_i, \cdots]$ and each sentence s_i is attributed to (or grounded in) a set of textual segments X_i retrieved from C. The report request contains the report requester's background, problem statement in a few sentences, and the length limit of the report (2000 characters).

The RAGTIME track includes three variants or subtasks of the task above.

- **Multilingual report generation**: Given a set of documents in multiple languages and a "report request" that describes the information need, generate a report in English that summarizes the information.

- **Monolingual English report generation**: This task is similar to Multilingual Report Generation, but only English documents are used.

- **Retrieval** (cross-language and multilingual): Given a set of documents in multiple languages and a "report request" that describes the information need, retrieve documents that help satisfy this information need.

This task formulation is very similar to that of the RAG track (cf. Section 2.1).

2.2.1 Datasets

RAGTIME track organizers provide the following datasets [32].

Text collection C: RAGTIME track organizers provide RAGTIME1 as the text collection [32]. The collection includes 4,000,380 news articles in four languages (Arabic, Chinese, English, and Russian) sampled from the Common Crawl News between August 1, 2021 and July 31, 2024. Each language has an equal number of 1,000,095 news articles. Each document has two fields: URL and text. The track organizers also provided English translations of the Arabic, Chinese, and Russian documents through a machine translation model.

Queries $\{q\}$: The development report requests ("topics" in TREC parlance) include those used in the pilot study of report generation, part of the 2024 TREC NeuCLIR track [29]. These report requests are complex questions that need to be answered by possibly synthesizing information from multiple documents. The track organizers did not provide document-level relevance judgments ("qrels" in TREC parlance) for these report requests. In 2025, RAGTIME organizers first released a set of dry run topics and then released a set of final topics.

An example report request is the following:

> **Background**: "I am a Hollywood reporter writing an article about the highest-grossing films, Avengers: Endgame and Avatar."
> **Problem Statement**: "The article needs to include when each of these films was considered the highest-grossing film and any manipulations undertaken to bring moviegoers back to the box office with the specific goal of increasing the money made on the film."
> **Limit**: 2000 characters.

Like the topic narratives used in the RAG track, RAGTIME report requests express complex, nuanced, and multifaceted information needs.

2.2.2 Evaluation methodology

The RAGTIME track organizers described their evaluation methodology, called *ARGUE* (Automated Report Generation Under Evaluation), in both the track guidelines [33] and a perspective paper published in SIGIR 2024 [13]. We refer the reader to these original sources for detailed explanation and examples of the evaluation methodology. Below, we summarize the main ideas underlying the ARGUE framework.

- ARGUE defines a "nugget" as a question relevant to an aspect of the report request and a set of corresponding answers properly grounded in documents from the collection. Conceptually, each nugget concerns a relevant aspect of the report request, in the form of a question and its answer(s). Stylistically, a nugget could be rendered as a heading or bullet that contains the question, followed by sentences or sub-bullets that contain the corresponding answers, each with citation link(s). When evaluating a generated report, the report covers a nugget if and only if it contains a sentence that answers the nugget question *and* that sentence cites one or more documents that properly support this answer.

  The set of nuggets is curated through a combination of a report requester's *a priori* brainstorming (before seeing any materials, what questions should be answered in a satisfying report?) and *a posteriori* validation (after seeing submitted reports and retrieved documents, what questions turn out to be useful in a satisfying report?). The set of nuggets aim to provide wide-enough coverage of relevant content, and is used to score a report's **nugget recall**.

- ARGUE differentiates multiple "needs for citation" in a report. Some sentences, such as introductory and summary sentences, do not need to have a citation. For sentences that successfully answer nugget questions, correct citations are rewarded, and incorrect or missing citations are penalized.[1] For sentences that carry substantive information but do not answer nugget questions, correct citations are neither rewarded nor penalized, but incorrect or missing citations are penalized. If a sentence states the absence of information and does not have a citation (rightfully so), and that information is indeed absent according to the curated nuggets, then that sentence should be rewarded. If, on the other hand, the information is indeed present in the curated nuggets, then that sentence should be penalized.

[1]A citation is correct (incorrect) if the sentence can (cannot) be grounded in the cited source, respectively.

This per-sentence rating then becomes the basis to evaluate **citation precision**: out of all sentences that concern the curated nuggets and therefore require citations (or non-citations in the case of information absence), what is the percentage of sentences that have correct citations or correct non-citations?

- ARGUE uses both nugget recall and citation precision as metrics to evaluate reports.

The evaluation methodologies of the RAG and RAGTIME tracks share many components: nugget-based evaluation, citation evaluation, and retrieval evaluation (implicit in RAGTIME/ARGUE). Different from the RAG track, the RAGTIME track chooses not to rate the fluency of a report.

3 Agentic RAG

Agentic RAG is a class of systems where an LLM (or a set of LLMs) operates as an agent that can autonomously plan its steps, iteratively retrieve information from external sources (such as databases, APIs, or search), evaluate and refine responses, and adapt its behavior across multi-turn tasks or long-horizon goals.

It enhances traditional RAG by embedding agentic capabilities such as:

- Decomposition of tasks (e.g., breaking a complex question into subtasks);
- Strategic retrieval (e.g., choosing which documents to fetch based on intermediate results);
- Self-reflection and iteration (e.g., revising a draft answer after identifying weak points);
- Tool use (e.g., calculators, code execution, or calling other APIs).

Agentic RAG is discussed in [9] as a method to improve retrieval and generation performance by allowing the system to make query and writing decisions. Practical implications of agentic systems include reducing system complexity, improving system security, and improving user experience, and are suggested in [25].

3.1 Agentic RAG Pipeline Design

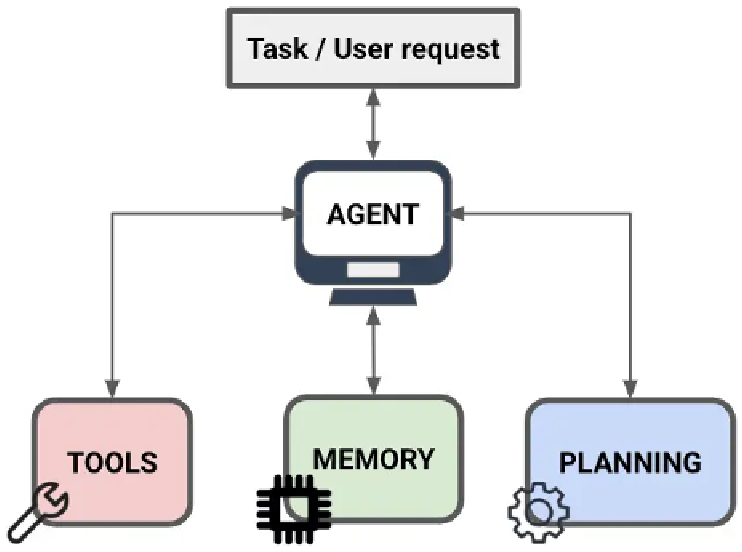


Figure 1: AI Agents [20]

Agents possess capabilities for tool use, memory, and planning, enabling fine-grained control over the search and generation process. Agents make decisions on behalf of the user and take resulting actions.

The Agentic RAG pipeline differs from conventional RAG by incorporating task decomposition, self-evaluation, and iteration. As illustrated in figure 2, it introduces components such as:

- **Query Analysis:** Determines if a query relates to personal or external data.
- **Retrieval Grading:** Classifies retrieved documents before passing them to the generator.
- **Recursive Planning:** If no relevant information is found, the query is rewritten and retried.

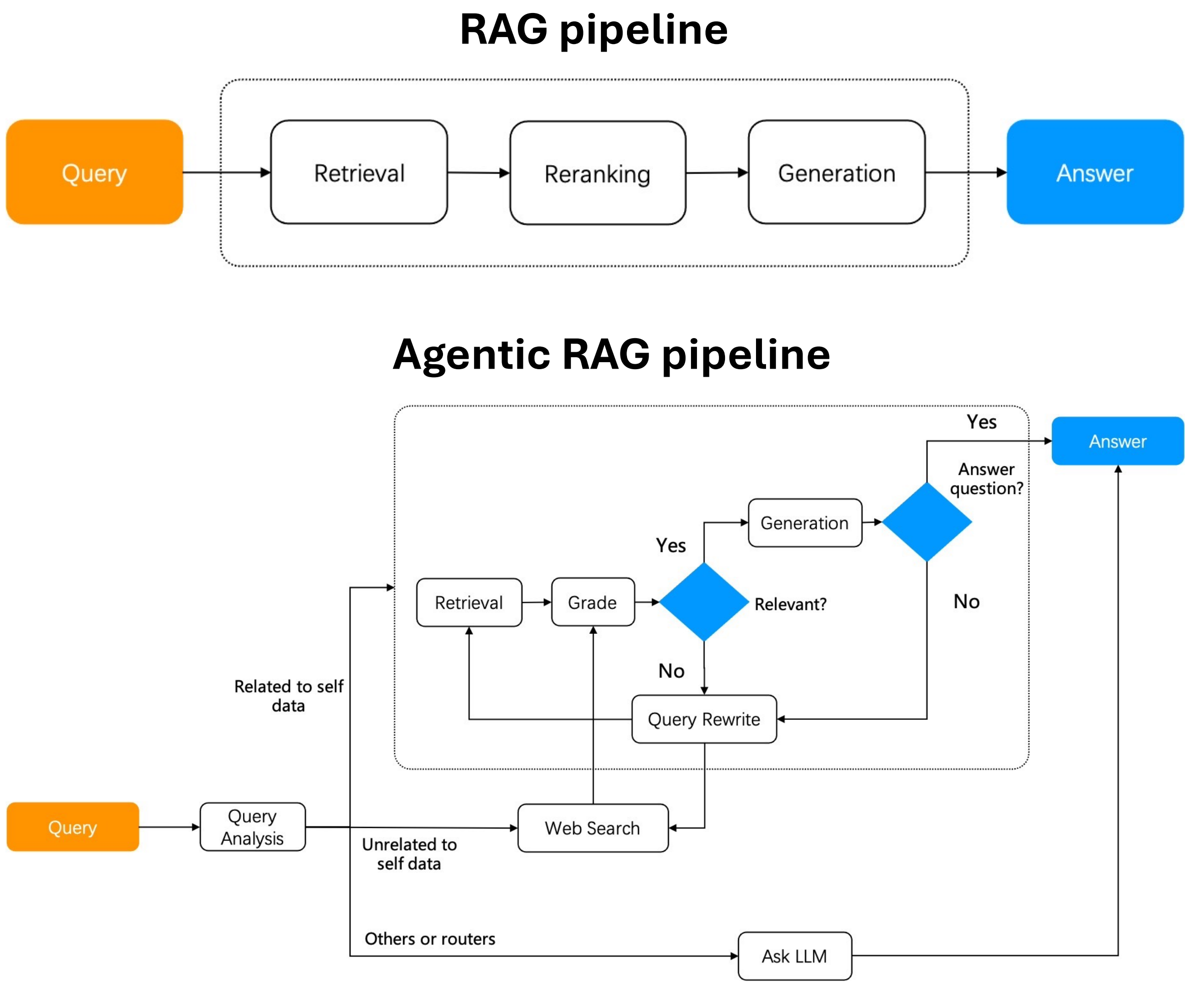


Figure 2: Agentic RAG Pipeline [23]

3.2 Agent Team Structures

Several team-based agent configurations were considered and implemented using Microsoft's Autogen framework [14]:

- **Single Agent:** Interprets user input, uses tool(s) to retrieve context, and generates an answer.
- **Directed Graph:** Planner → Research Assistant → Writer.

- **Swarm:** Planner, Research Assistant, Writer, Reviewer (fully connected).
- **Selector Group Chat:** Selector decides on best path among agents.
- **Round Robin:** Sequential handoff across agents.

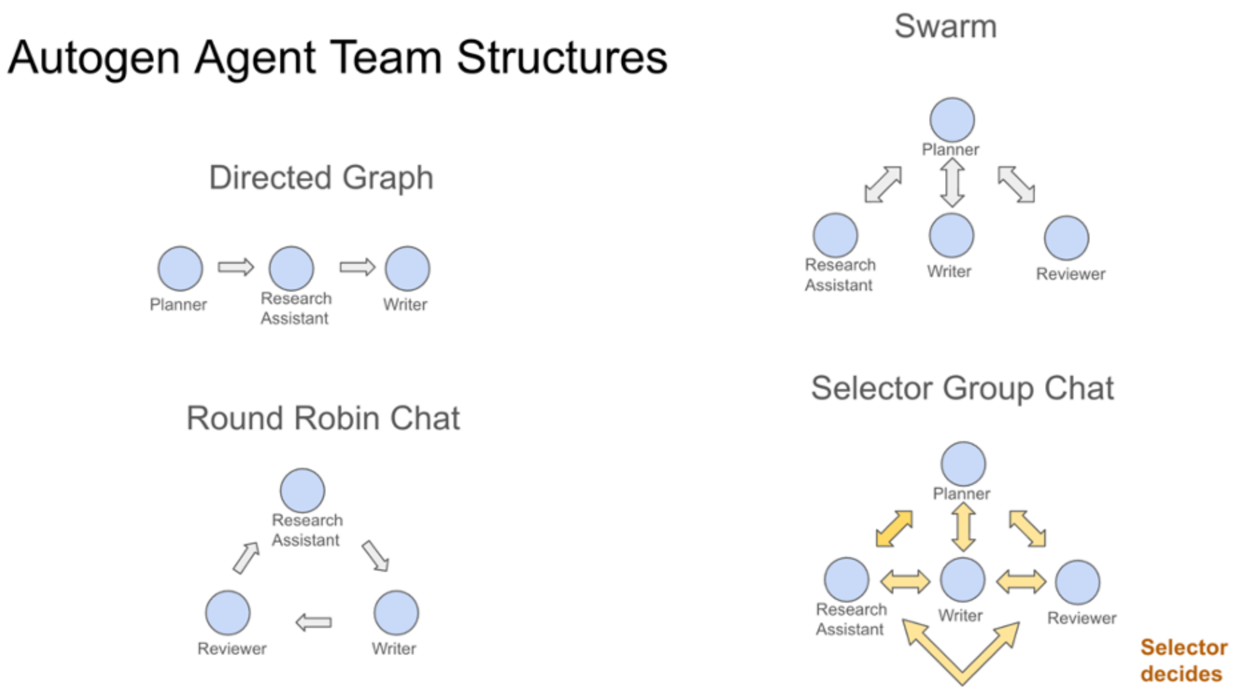


Figure 3: Microsoft Autogen Agent Team Structures

Each agent is defined by a system prompt providing instructions for the agent's role. The planner decomposes the input task, creates a research plan, and assigns tasks to a research assistant, a writer, or a reviewer. The agents also have access to retrieval tools to retrieve relevant context. Based on the retrieved context, the research assistant is instructed to compose a list of relevant information. The writer then composes an answer for output, citing the source context where appropriate. The reviewer is instructed to check an answer for completeness and for citation consistency.

4 RAG Track

4.1 Retrieval Submissions

4.1.1 Index Construction

Sparse Word Representation + Inverted Index: BM25. BM25 is a classical method in ad-hoc retrieval. First introduced in the early 1990s, it has been a strong contender, especially when dealing with keyword queries [22].

We used Apache Solr to process the segment field of MSMARCO V2.1 as field type `text_general`, build inverted index for all 13,520,750 segments, and use the default BM25 similarity function to rank segments for a given query. Since BM25 is known to work less well with queries that contain stopwords, we remove stopwords from the query. We used the stopword list in the NLTK package. No stemming was performed on either the query or the segment text.

Sparse Learned Representation + Inverted Index: SPLADEv3. SPLADE (Sparse Lexical and Expansion Model) [7] is a family of trainable sparse text representation methods that represent a piece of text by expanding each token into a set of most probable alternative tokens as predicted by an encoder transformer. The relevance score is a dot product between the query sparse vector and the document sparse vector. The main advantage of using sparse representations is that we can reuse the highly efficient inverted index machinery implemented in Apache Solr.

We used the SPLADEv3 model released by the Naver Research Lab on Huggingface [6]. SPLADEv3 is a recent SPLADE model released in 2024. Compared to the original SPLADE model, SPLADEv3 was trained using much harder negative examples and distilled knowledge from a larger ensemble of teacher models. The encoder has a maximum input length of 512 tokens. The model sizes are both about 110M parameters, the same size as the BERT-base model. We configured the output to be in Anserini format [24] and used Apache Solr/Lucene to construct inverted index for the sparse document vector (treating subwords as non-analyzed strings). We adopted Anserini's impact similarity[2] to implement the dot product scoring in Solr.

4.1.2 Nugget-Augmented Retrieval

A known approach to improving retrieval performance is query refinement. Query refinement works by framing better questions and using more accurate keywords as search queries. Traditionally, query refinement techniques include (1) using a thesaurus to look up synonyms, (2) using query reformulation logs (if available) to expand the original query, (3) using corpus-level word co-occurrence statistics to obtain related terms, and (4) using pseudo-relevance feedback to harvest related terms from top-ranked results. In this study, we combine these ideas through a generation step by leveraging a large language model and top-ranked results.

Specifically, we take top-ranked n documents returned by the original query, and use GPT-4o to summarize these documents into a set of "nuggets." Each "nugget" is an atomic content unit relevant to the query, and different nuggets do not overlap in content [19]. These nuggets are rated as either "vital" or "okay" depending on their relevance to the original query. These nuggets are then used to formulate new queries and search the index again. We use the implementation in `castorini/nuggetizer`[3] for nugget generation.

We evaluated nugget-augmented retrieval on the 301 complex queries and relevance judgments from the 2024 TREC RAG track. We used the nuggets in two ways:

- **Concatenating nuggets**. This method retrieves results using the original complex query, generates nuggets using the top retrieved results, and then concatenates the original query and multiple nuggets into one big query and searches the index again.
- **Separately searching nuggets**. This method retrieves results using the original complex query, generates nuggets using the top retrieved results, and then concatenates the original query and that nugget, and uses each of these expanded queries to search again. Multiple search result lists are then merged into one list using reciprocal rank fusion.

Our preliminary experiments found that only using "vital" nuggets in the above procedure is always better than using both "vital" and "okay" nuggets in terms of retrieval performance metrics. Therefore, only "vital" nuggets were used.

[2]`https://github.com/castorini/anserini/blob/master/src/main/java/io/anserini/search/similarity/ImpactSimilarity.java`

[3]`https://github.com/castorini/nuggetizer`

Table 1: Performance of different retrieval methods averaged over the 301 topics (queries) used in the 2024 TREC RAG track. BM25+ and SPLADE-v3+ are BM25 and SPLADEv3 augmented by concatenating nuggets; BM25++ and SPLADE-v3++ are BM25 and SPLADE-v3 augmented by separately searching nuggets. We lightly tuned the parameter $k = \{5, 10, 30, 50\}$ in reciprocal rank fusion and picked the best-performing one. In BM25++, $k = 30$. In SPLADE-v3++, $k = 10$.

Method	**Mean reciprocal rank**	**Precision@10**	**Recall@100**	**NDCG@100**
BM25	0.38	0.21	0.12	0.20
BM25+	0.52	0.30	0.16	0.26
BM25++	0.51	0.30	0.17	0.27
SPLADEv3	0.61	0.43	0.29	0.46
SPLADEv3+	0.70	0.48	0.29	0.42
SPLADEv3++	0.67	0.51	0.32	0.48

Table 1 shows that nugget-augmented retrieval does improve the performance of the corresponding base method. On the mean reciprocal rank metric, concatenating nuggets (+) is better than separately searching nuggets (++). However, on other metrics (precision@10, recall@100, NDCG@100), separately searching nuggets (++) is better than concatenating nuggets (+).

4.1.3 Submitted Runs

Our four retrieval submissions used the following setup, two of which employed nugget generation:

- **Concatenating subquestions** (runtag: `LAS_con-que`): Use GPT-4o to rephrase the original narrative into a set of questions. Concatenate these questions into a big query. Search SPLADEv3 segment index using the big query.
- **Separately searching subquestions** (runtag: `LAS_sep-que`): Use GPT-4o to rephrase the original narrative into a set of questions. Search each of these questions in SPLADEv3 segment index. Merge result list using reciprocal rank fusion ($k = 10$).
- **Concatenating subquestions then concatenating nuggets** (runtag: `LAS_con-que-con-nug`): Use GPT-4o to rephrase the original narrative into a set of questions. Concatenate these questions into a big query. Search SPLADEv3 segment index using the big query. Use `castorini/nuggetizer` to generate nuggets based on the top $n = 20$ results. Nugget creation prompt was modified to generate self-contained 10-20 words nuggets. Concatenate only "vital" nuggets into a big query. Search SPLADEv3 segment index using the big query.
- **Concatenating subquestions then separately searching nuggets** (runtag: `LAS_con-que-sep-nug`): Use GPT-4o to rephrase the original narrative into a set of questions. Concatenate these questions into a big query. Search SPLADEv3 segment index using the big query. Use `castorini/nuggetizer` to generate nuggets based on the top $n = 20$ results. Nugget creation prompt was modified to generate self-contained 10-20 words nuggets. Search each "vital" nuggets in SPLADEv3 segment index. Merge result list using reciprocal rank fusion ($k = 10$).

Note that our RAG submissions (Section 4.2) used the plain BM25 and SPLADEv3 indices as retrieval tools without nuggetization. Query decomposition and results merging are handled in agentic RAG workflows.

4.2 RAG Submissions

4.2.1 Method

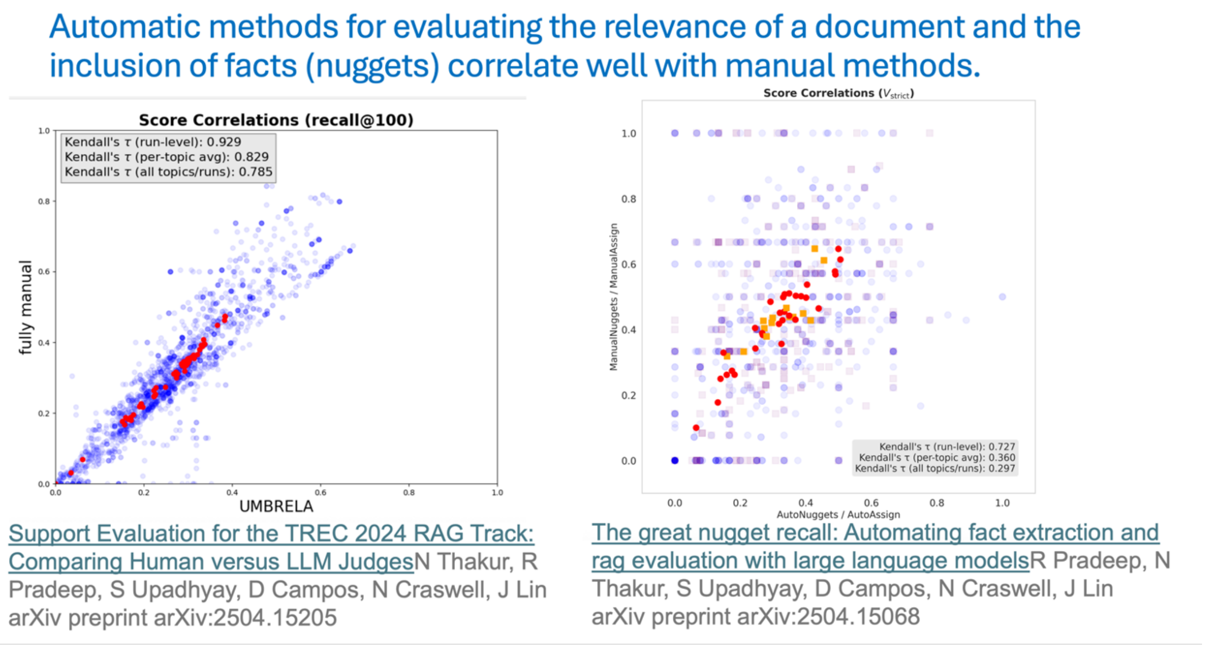
We used Microsoft's Autogen agent framework to compose agents and team structures as outlined previously. As mentioned above, we created multiple search indices to retrieve MSMarco document segments. These include a BM25 text index, a SPLADEv3 index, and a dense vector index built with T5 embeddings and a qdrant vector database. We built a Model Context Protocol (MCP) server that provides multiple tools to our agents, allowing them to query segments in these indices and retrieve full document text.

4.2.2 Evaluation Metrics

In order to measure system performance, we employed Open-RAG-Eval [35], which includes:

- **UMBRELA:** Precision-based retrieval metric that uses an LLM judge to give integer scores to retrieved documents from 0 (not relevant) to 3 (fully answers question) [12].
- **Autonuggetizer:** For evaluating nugget coverage (fact recall), this metric prompts an LLM to list relevant pieces of information in the retrieved context. Then it checks if each piece of information is represented in the RAG output [19].
- **Vectara's Hughes Model:** Hallucination detection and factuality scoring using a fine-tuned transformer model that compares the text in the retrieved context with the text in the RAG output. 0 means the answer does not represent the context and is hallucinated. 1 indicates that all facts in the generated output accurately reflect the context.
- **Citation F1:** Measures citation relevance, where precision is the average score (0=no relationship, 0.5=partial relationship, 1=strong relationship as evaluated by an LLM as a judge) across citations the model made and recall is an estimated coverage measure, calculated as the average citation score across all parts of the answer.

These evaluation methodologies were developed by the sponsors of the TREC RAG track and used in contest evaluations. The UMBRELA retrieval metric and the Autonuggetizer process were shown to correlate with human judgment, as shown in Figure 4.



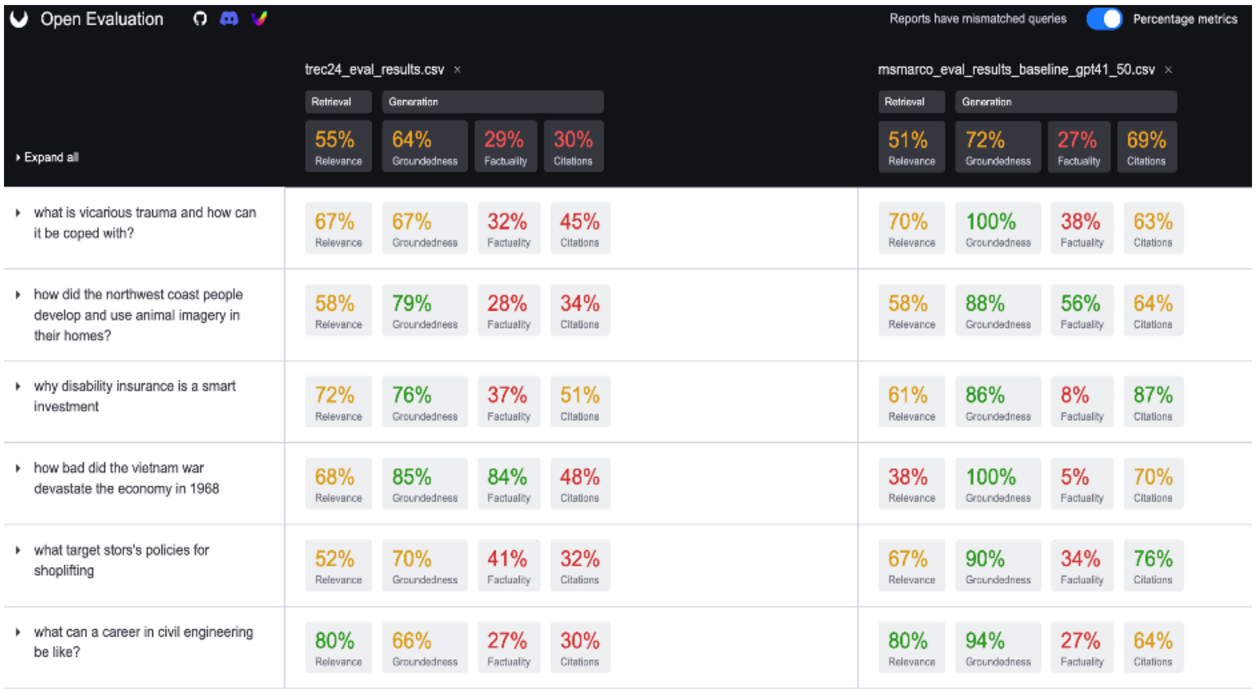
Automatic methods for evaluating the relevance of a document and the inclusion of facts (nuggets) correlate well with manual methods.

Support Evaluation for the TREC 2024 RAG Track: Comparing Human versus LLM Judges N Thakur, R Pradeep, S Upadhyay, D Campos, N Craswell, J Lin arXiv preprint arXiv:2504.15205

The great nugget recall: Automating fact extraction and rag evaluation with large language models R Pradeep, N Thakur, S Upadhyay, D Campos, N Craswell, J Lin arXiv preprint arXiv:2504.15068

Figure 4: UMBRELA and Autonuggetizer Correlations with Human Judgement

The Open-RAG-Eval package produces an output file that can be uploaded into a web viewer in order to inspect RAG outputs and evaluation results. See Figure 5.



	trec24_eval_results.csv				msmarco_eval_results_baseline_gpt41_50.csv			
	Retrieval	Generation			Retrieval	Generation		
	55% Relevance	64% Groundedness	29% Factuality	30% Citations	51% Relevance	72% Groundedness	27% Factuality	69% Citations
what is vicarious trauma and how can it be coped with?	67% Relevance	67% Groundedness	32% Factuality	45% Citations	70% Relevance	100% Groundedness	38% Factuality	63% Citations
how did the northwest coast people develop and use animal imagery in their homes?	58% Relevance	79% Groundedness	28% Factuality	34% Citations	58% Relevance	88% Groundedness	56% Factuality	64% Citations
why disability insurance is a smart investment	72% Relevance	76% Groundedness	37% Factuality	51% Citations	61% Relevance	86% Groundedness	8% Factuality	87% Citations
how bad did the vietnam war devastate the economy in 1968	68% Relevance	85% Groundedness	84% Factuality	48% Citations	38% Relevance	100% Groundedness	5% Factuality	70% Citations
what target stors's policies for shoplifting	52% Relevance	70% Groundedness	41% Factuality	32% Citations	67% Relevance	90% Groundedness	34% Factuality	76% Citations
what can a career in civil engineering be like?	80% Relevance	66% Groundedness	27% Factuality	30% Citations	80% Relevance	94% Groundedness	27% Factuality	64% Citations

Figure 5: Open RAG Eval Results Viewer

4.2.3 TREC 2024 Experiments

Table 2: TREC 2024 Test Results

Model	Relevance	Groundedness	Factuality	Citation F1	Avg Docs Retr.	Answer Avg Len	Avg No. Citations
LAS 2024 results	55	64	29	30	20.00	302.48	15.56
Baseline GPT-4.1 Agent	51	72	27	69	15.41	298.26	9.65
Digraph Team	43	69	28	60	27.86	185.68	4.76
Swarm Team	46	65	37	58	15.22	256.26	5.54
Selector Team	59	63	41	63	6.50	192.38	7.84

Here we use the first 50 examples from the 2024 TREC RAG test set to evaluate agentic system performance and compare against our best previous year's submission. See [37] for details on our 2024 submissions. The baseline agent was instructed to decompose the query if necessary and use both BM25 and Splade retrieval tools. It then wrote an answer citing the retrieved context. A GPT-4.1 model was used for the agent.

Inspired by hypothetical RAG [8], the digraph team had the first agent perform a quick BM25 search and compose an answer. The research assistant then decomposed the answer into questions and queried the SPLADE index. The relevant results were compiled into a list by the research assistant and passed to the writer. The writer composed the final answer. No reviewer was used. All agents used GPT-4o-mini.

The swarm team used a planner with GPT-4.1 to create a research plan and hand off tasks to the other team members. The research assistant only queried the SPLADE index and produced a list of relevant information. A writer agent produced a final answer based on the relevant information provided. Each agent's turn began with a handoff from the planner and ended with the agent handing results back to the planner.

The selector group chat team used GPT-4.1 for all agents. Starting with a query, a selector or router would direct it to the planner. The planner would analyze the input and produce a plan. The selector would then choose the next agent in turn based on the plan and/or the previous agent's output. In addition to the planner, the team consisted of a research assistant, a writer, and a reviewer. The research assistant and writer acted just as in previous team examples. A reviewer was included to test whether it would spur additional research or writing iterations; however, this behavior was observed only seldomly.

Relevance and Citation accuracy scores were improved in the selector group chat team example because we started using `miniblame` for citation attribution. The `miniblame` package uses semantic similarity at the sentence level to find likely supporting sentences in the documents. This package was based on previous work on a human evaluation of natural language inference (NLI) and semantic similarity for attribution [1] [4] Only cited documents were considered retrieved.

4.2.4 TREC 2025 Experiments

Here we use the first 25 examples from the 2025 TREC RAG test set. Input queries changed significantly from TREC 2024 RAG to the 2025 edition. For example, the first topic from 2024 is *"What is vicarious trauma and how can it be coped with?"* The 2025 queries more closely resemble a research task that might be posed to OpenAI Deep Research, for example:

> *"I want to understand how ethical theories like utilitarianism and Kantianism, along with professional codes, influence decision-making and leadership in organizations and professions such as social work and teaching. How do these ethical standards impact hiring, leadership styles, and personal behavior? I'm also puzzled why ethical people might still make unethical choices."*

This change suggests that the query decomposition and iterative approaches afforded by our agentic systems might be even more consequential.

Table 3: TREC 2025 Test Results

Model	**Relevance**	**Groundedness**	**Factuality**	**Citation F1**	**Answer Avg. Len**	**Avg Time/Answer (s)**	**Prompt tokens**	**Completion tokens**
Single Agent	60	84	24	70	543.64	51.16	422682	30154
Selector Team	61	83	25	70	358.76	75.36	417856	49737
Digraph Team	61	86	16	67	429.64	141.61	399668	62105
Swarm Team	59	82	35	62	635.84	154.66	3778378	143594

Here, we used `miniblame` attribution to assign citations and pick relevant segments.

The Selector Team scores reflect the use of GPT-4.1 for each agent: planner, research assistant, writer, and reviewer.

The Digraph Team used the reasoning model GPT-4o-mini for the planner, and GPT-4.1 for the research assistant and writer. No reviewer was used.

The Swarm team revised the planner agent prompt to force a workflow with a second round of research. The planner was also instructed to play a more active role between agent turns, compiling previous results and giving more explicit instructions to other agents. GPT-4.1 was used for the planner and GPT-4o-mini for the research assistant and writer.

In each case above, both the planner (if present) and the research assistant had the ability to decompose or otherwise suggest retrieval queries. Only context segments from the SPLADE index were retrieved.

While we don't see a large performance difference between team constructs according to most of our metrics, we do see a large difference in time for the system to return an answer and the total tokens used.

4.3 Results

4.3.1 Retrieval Only

From our retrieval results, we have some unexpected and interesting conclusions. Concatenating subquestions into one big query (LAS_con-que) works better than searching each subquestion and then fusinig results through reciprocal rank fusion (LAS_sep-que). Also concatenating nuggets into one big query (LAS_con-que-con-nug) works better than searching each nugget and then fuse results through reciprocal rank fusion (LAS_con-que-sep-nug).

[4]This evaluation used the `sentence-t5-xxl` model [16]. Still, we used `sentence-t5-base` as [3] found the smaller model sufficed for the neuCLIR Pilot data from TREC 2024.

Run	NDCG@100	NDCG@30	Recall@100
LAS_con-que-con-nug	0.4861	0.6692	0.1906
LAS_con-que-sep-nug	0.4382	0.6446	0.1672
LAS_con-que	0.4862	0.5545	0.2072
LAS_sep-que	0.3880	0.5332	0.1550
Avg. per-query min	0.0923	0.1605	0.0209
Avg. per-query medium	0.4345	0.5423	0.1732
Avg. per-query max	0.6536	0.7732	0.2832

Table 4: Evaluation results across runs and per-query statistics

At least for our retriever (splade-v3), using one big query (concatenating subquestions or nuggets) work better than fusing per-query results. This is different from preliminary findings on TREC 2024 topics, where fusing results gave better performance than one big query.

Concerning one round vs. two rounds of query rewriting and search, when we compare LAS_con-que against LAS_con-que-con-nug, it seems that using nuggets to do another round of search (essentially two rounds of LLM-based query rewriting) may help improve precision (ndcg_cut_30) but not so much for recall (ndcg_cut_100, recall_100).

This result makes sense as the first round of query rewriting may have already produced enough useful query terms, and the marginal gain from a second round is minimal. A second around of query rewriting and retrieval required a lot of time and compute, so it is not worth doing.

In summary, we can conclude LAS_con-que is the "best bang for the buck" among our four retrieval runs. It simply uses GPT-4o to decompose (rewrite) the provided narrative paragraph into a set of subquestions, concatenate them as one big query, and search splade-v3 index only once.

4.3.2 Retrieval Augmented Generation

The team submitted two full RAG runs (generations for all 105 questions). The first submission was generated in a pipeline using a single agent only, and the second submission was generated in a pipeline using the Selector Team described above. Initial results as seen in Figure 6 demonstrate our methods were quite successful with regards to subject coverage, as we matched the top score for 10 of 20 topics evaluated. We also performed consistently above the median for strict vital nugget recall. See Figure 7. Somewhat surprisingly, the single agent performed better than the multi-agent system with our configurations.

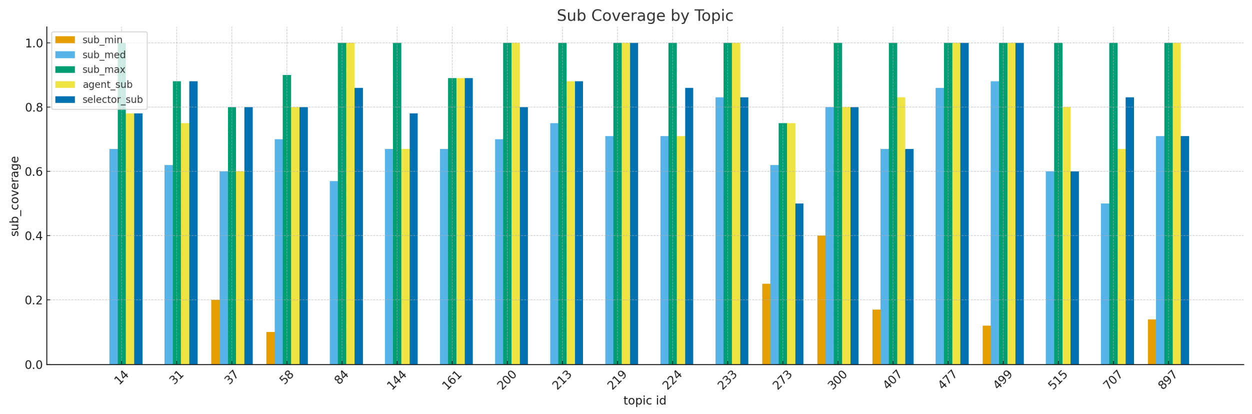


Figure 6: Subject Coverage Scoring Results

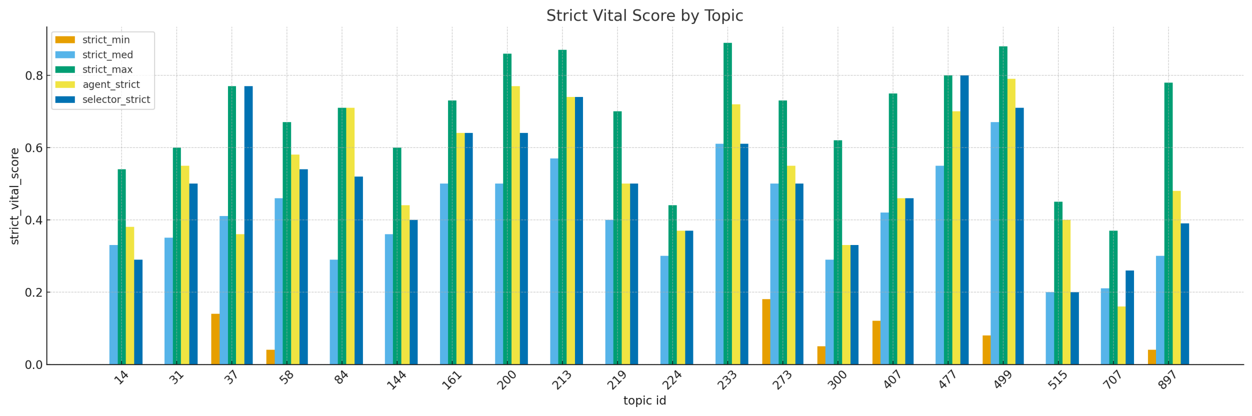


Figure 7: Strict Vital Scoring Results

The following figures show that compared to other submissions, LAS RAG runs were quite successful in strict vital nugget evaluation and subject coverage, but achieved middling citation accuracy results.

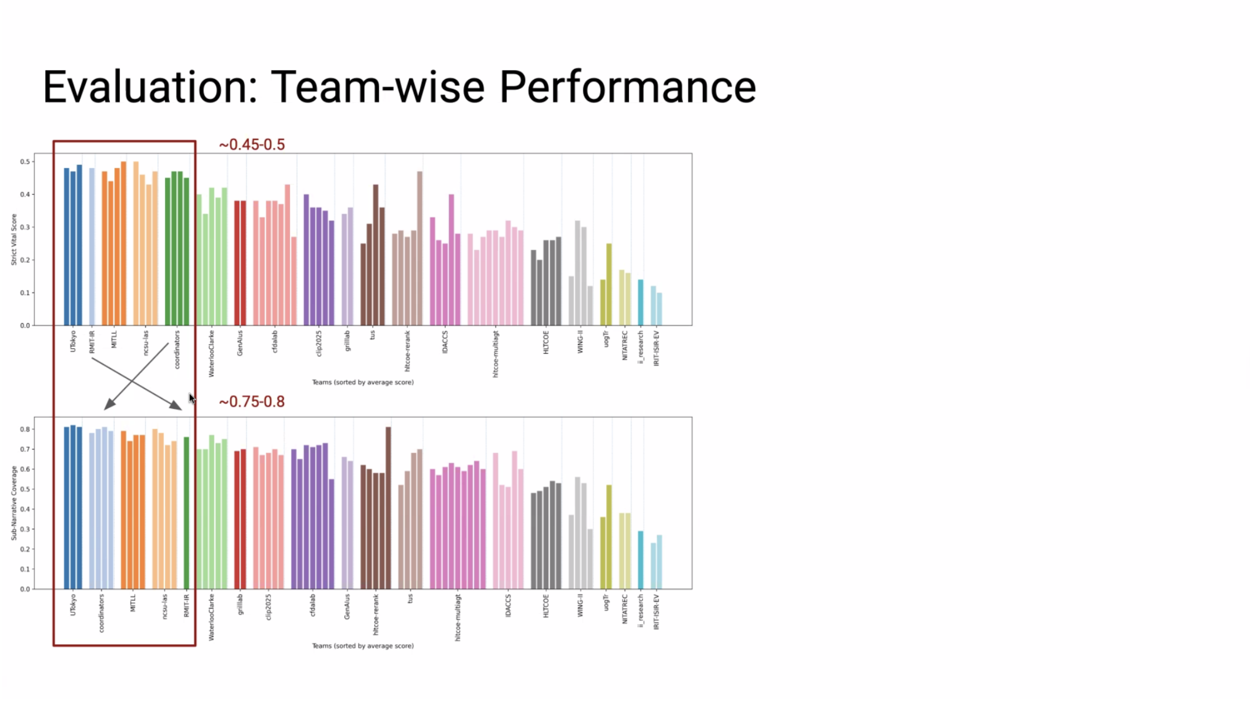


Figure 8: Strict Vital and Subject Coverage Team Comparison

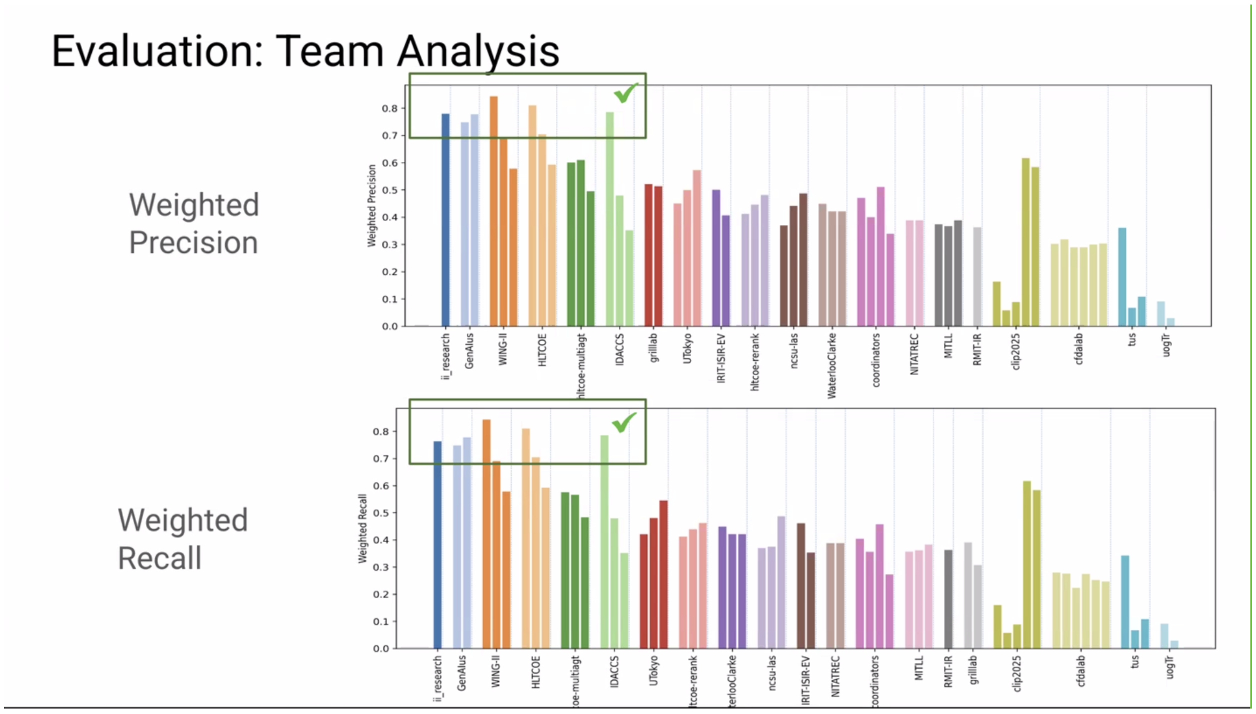


Figure 9: Citation Results Team Comparison

5 RAGTIME Track

In this section, we briefly describe our work on adapting some of the agentic workflows developed for the TREC RAG track to RAGTIME. Here, we will mostly discuss very high-level strategies. We will also discuss the differences in the pipeline. The agentic RAGTIME submissions were also considerably more costly to produce than our hybrid, paraphrased-occam's system as described in [4]. This is true across many dimensions: wall-clock time, compute time, and monetary cost for API calls to OpenAI. Our experience is that these agentic workflows can be very powerful, but they can also be fragile and expensive.

5.1 A Different Search Service

To help the RAGTIME track participants, the track organizers provided a document search and retrieval service for the track's multilingual corpus. Details can be found on the track website `https://trec-ragtime.github.io/search_api.html`. It uses PLAID-X, a very effective cross-lingual information retrieval algorithm [38]. This was immensely helpful, and we used it almost exclusively for our document search as well as for retrieving the text of the documents. It supported both the original languages and the machine-translated documents. We had systems that used the original, the translations, and both.

5.2 Tools and Techniques

We experimented with three different agentic workflows: a simple single agent, a round robin workflow, and one using a selector agent. The simple single agent looked very much like our submission to the RAG track in 2024. It used a single agent to do the bulk of the work. It did have the ability to use a search service as a tool. After that, it simply wrote the report and added references. These references were then checked and, in some cases, overwritten with `miniblame`.

5.3 Open RAG Eval

The python package `open-rag-eval`, open-sourced by Vectara, is designed to help measure the performance of RAG systems. The below quote is directly from their documentation:

> Importantly, open-rag-eval's metrics do not require "golden" nuggets or answers, making RAG evaluation easier and more scalable. This is achieved by utilizing UMBRELA and AutoNuggetizer, techniques originating and researched in Jimmy Lin's lab at UWaterloo.

This Waterloo group organizes the TREC RAG track. They also attempted to coordinate with the RAGTIME track organizers to standardize aspects such as submission file formats, making systems more likely to be useful for both tracks. Open RAG Eval gives several metrics. Some are based on retrieval, like relevance. Others are based on generative content. Were the desired facts in the report? They also had a hallucination/factuality metric. Also, importantly, they have metrics that measure the quality and reliability of attribution.

5.4 Comparative Analysis of Hybrid and Agentic Systems

Two examples are shown below. The first figure shows a comparison of content (often called nuggets in the Information Retrieval literature) across seven systems.

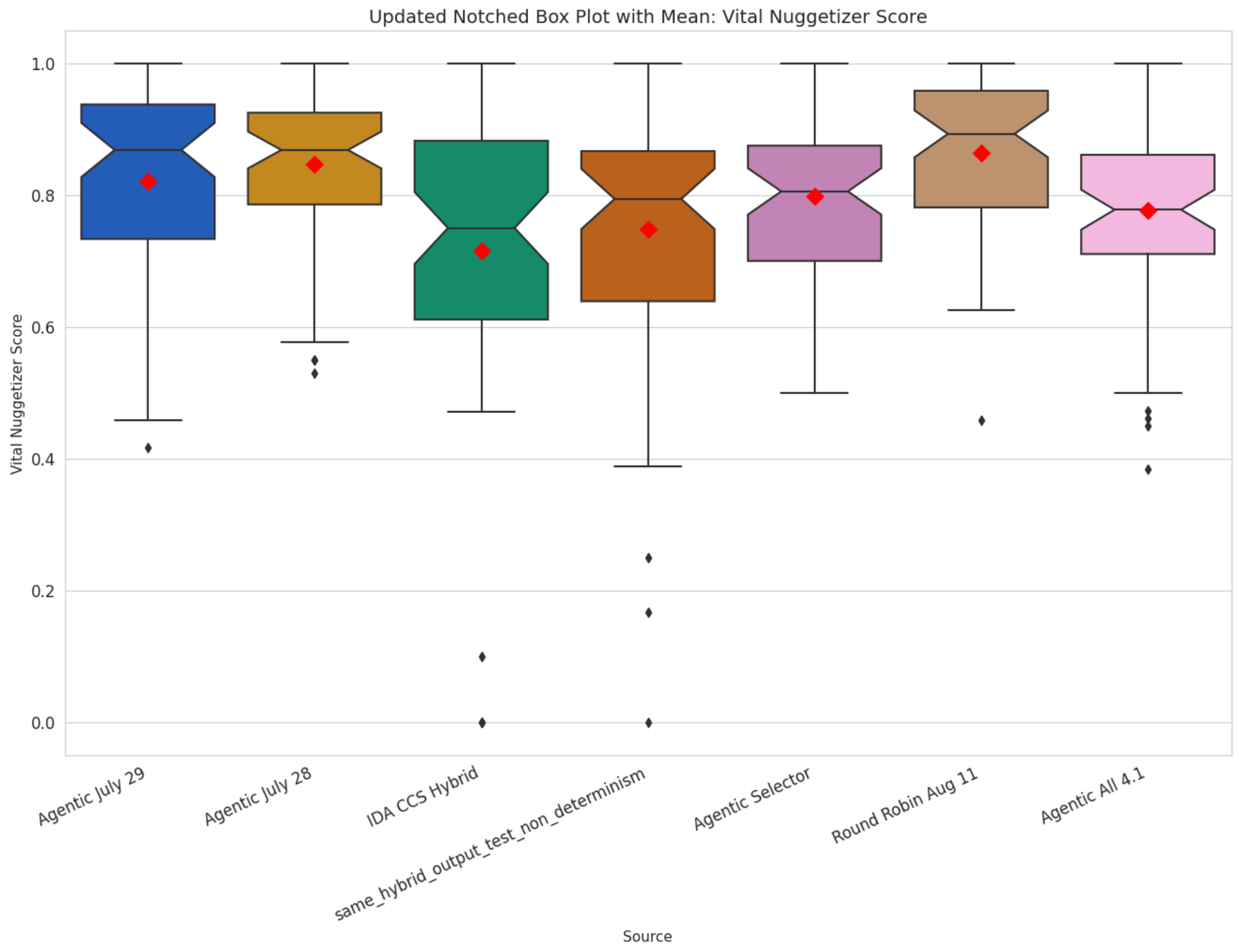


Figure 10: Comparison of Content Scores Across Systems

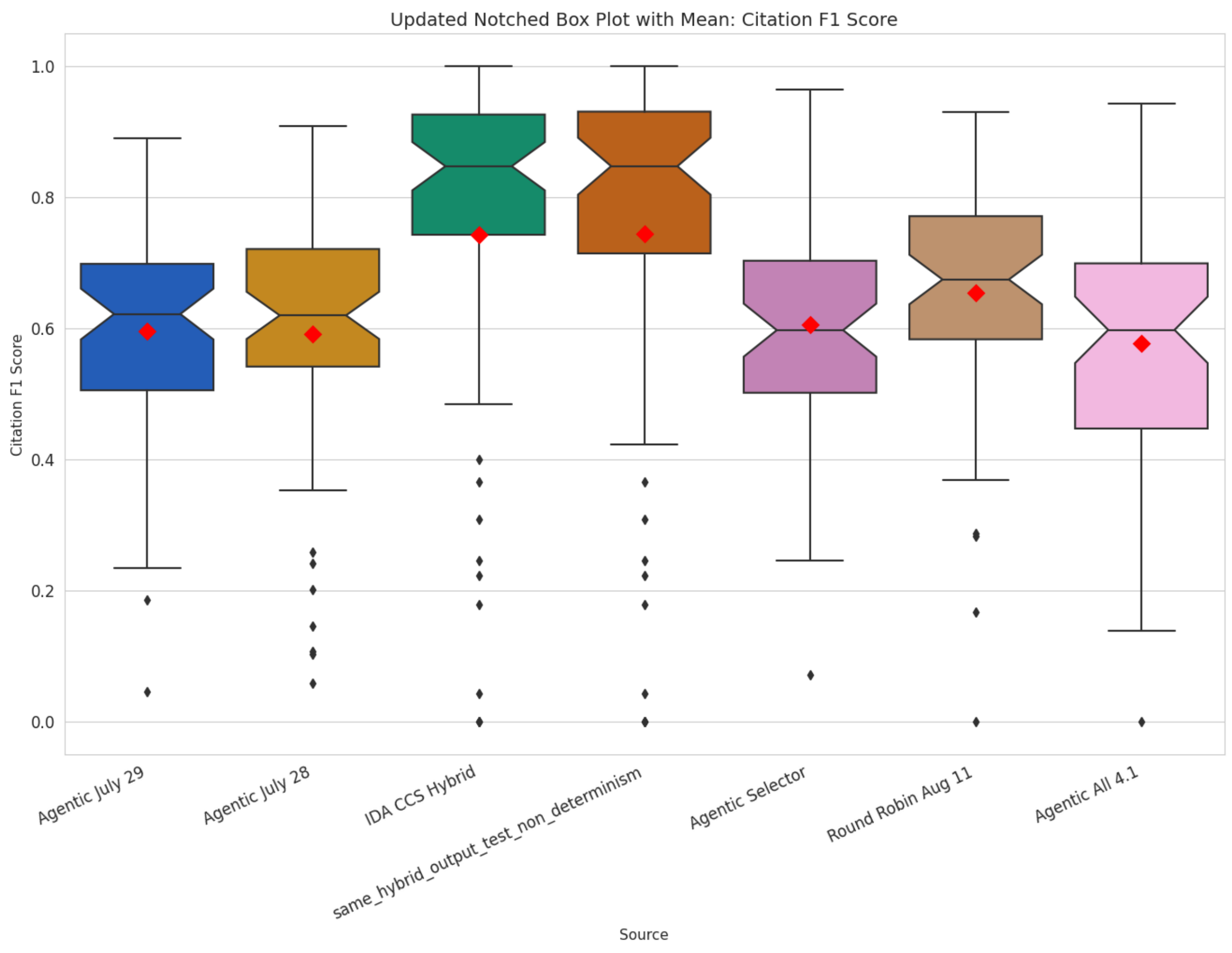


Figure 11: Comparison of Attribution Scores Across Systems

Five of them are based on agentic computations, and two of them were `occams` based hybrid summarization. Generally, as previous work has demonstrated, a hybrid does quite well on attribution. In this instance, the agentic methods tend to outperform in content as measured by Open RAG Eval.

In Figure 10, it is also noteworthy that the "IDA CCS Hybrid" and "same hybrid output test non-determinism" notched box-plots are for the same report/submission file. For the nuggetizer score, there are a couple of points difference at the level of the mean or median. Less so for the attribution one. This matches the previous discussion on non-determinism. It is mitigated at the level of averages and medians, but is still present.

5.5 Final Results

As with the RAG task submissions, LAS agentic RAGTIME submissions scored quite well in comparison to other teams with regards to nugget coverage, but was not as strong in other metrics as computed by the RAGTIME track's AUTOARGUE evaluations. See Figure 12 for example report generation results on long topics. Results for short topics were similar.

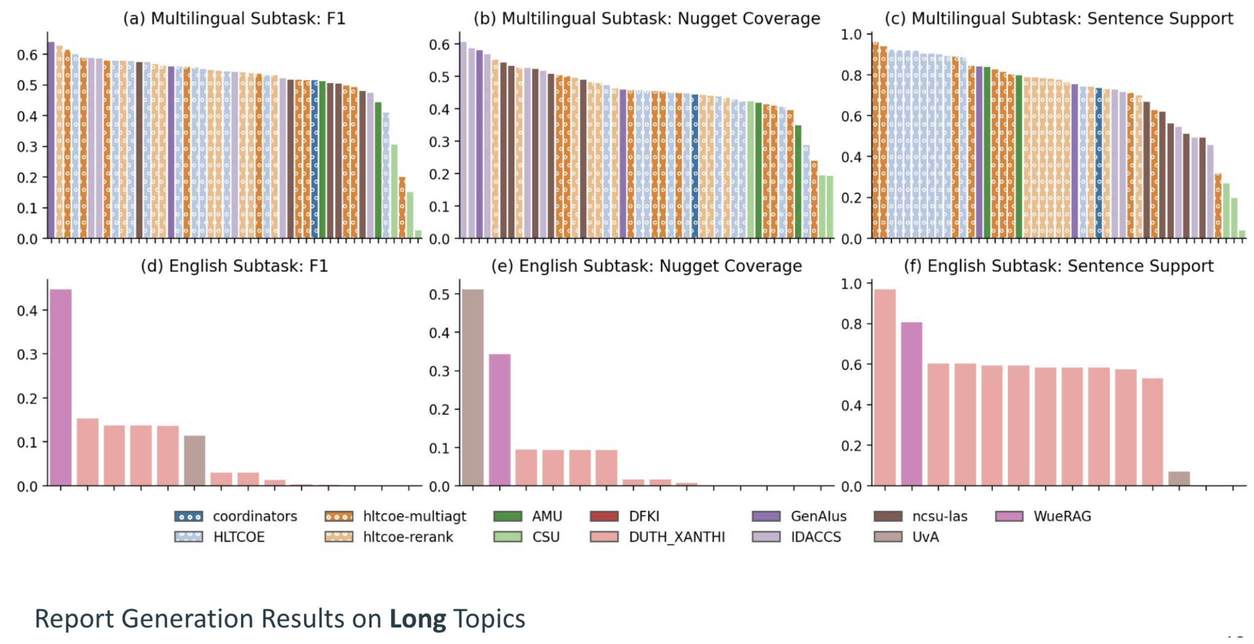


Figure 12: RAGTIME Team Performance Comparison

References

[1] Violet B, John M. Conroy, Sean Lynch, Danielle M, Neil P. Molino, Aaron Wiechmann, and Julia S. Yang. Where did you get that? Towards summarization attribution for analysts. *arXiv preprint arXiv:2511.08589*, 2025.

[2] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45, 2024.

[3] John M. Conroy, Razieh Fathi, Daria Symlova, and Y. Kelly Wu. Cross-lingual hybrid and abstractive summarization with automatic attribution for the neuCLIR 2024 pilot report generation track. The Thirty-Third Text REtrieval Conference (TREC 2024), Gaithersburg, MD, USA, November 15-18, 2024, 2024.

[4] John M. Conroy, Mike Green, Neil P. Molino, Yue "Ray" Wang, and Julia S. Yang. Team IDACCS at TREC 2025: RAG and RAGTIME tracks. In *Proceedings of the 34th Text REtrieval Conference (TREC 2025)*, Gaithersburg, Maryland, USA, 2025. National Institute of Standards and Technology (NIST). Notebook paper.

[5] Contributors to GitHub Anserini Repository. Anserini: BM25 baselines for the MS MARCO v2 collections. `https://github.com/castorini/anserini/blob/master/docs/experiments-msmarco-v2.md`. Accessed: 2024-07-22.

[6] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. From distillation to hard negative sampling: Making sparse neural IR models more effective. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*, pages 2353–2359, 2022.

[7] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. Splade: Sparse lexical and expansion model for first stage ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2288–2292, 2021.

[8] L. Gao, X. Ma, J. Lin, and J. Callan. Hypothetical document embeddings (HyDE) for zero-shot dense retrieval. *arXiv preprint arXiv:2212.10496*, 2022.

[9] Y. Gao et al. A survey of agentic RAG. *arXiv preprint arXiv:2501.09136*, 2025.

[10] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[11] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, 2022.

[12] J. Lin, S. Upadhyay, P. Narayan, S. Chawla, P. Pradeep, D. Campos, and N. Craswell. UMBRELA: A unified, multi-dimensional benchmark for rag evaluation. *arXiv preprint arXiv:2406.06519*, 2024.

[13] James Mayfield, Eugene Yang, Dawn Lawrie, Sean MacAvaney, Paul McNamee, Douglas W Oard, Luca Soldaini, Ian Soboroff, Orion Weller, Efsun Kayi, et al. On the evaluation of machine-generated reports. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1904–1915, 2024.

[14] Microsoft. Autogen. `https://microsoft.github.io/autogen/stable/index.html`, 2024.

[15] National Institute of Standards and Technology (NIST). Call for participation: Text retrieval conference (TREC) 2024. `https://trec.nist.gov/pubs/call2024.html`. Accessed: 2024-07-22.

[16] Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith B Hall, Daniel Cer, and Yinfei Yang. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. *arXiv preprint arXiv:2108.08877*, 2021.

[17] Arnold Overwijk, Chenyan Xiong, and Jamie Callan. ClueWeb22: 10 billion web documents with rich information. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*, pages 3360–3362, 2022.

[18] Ronak Pradeep, Nandan Thakur, Sahel Sharifymoghaddam, Eric Zhang, Ryan Nguyen, Daniel Campos, Nick Craswell, and Jimmy Lin. Ragnarök: A reusable RAG framework and baselines for TREC 2024 retrieval-augmented generation track. *arXiv preprint arXiv:2406.16828*, 2024.

[19] Ronak Pradeep, Nandan Thakur, Shivani Upadhyay, Daniel Campos, Nick Craswell, Ian Soboroff, Hoa Trang Dang, and Jimmy Lin. The great nugget recall: Automating fact extraction and RAG evaluation with large language models. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 180–190, 2025.

[20] Ai agents. `https://www.promptingguide.ai/agents/components`, 2024. Accessed: 2024.

[21] Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331, 2023.

[22] Stephen Robertson, Hugo Zaragoza, et al. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2009.

[23] Armand Ruiz Gabernet. Forget RAG and welcome agentic RAG. `https://community.ibm.com/community/user/blogs/armand-ruiz-gabernet/2025/01/15/forget-rag-and-welcome-agentic-rag`, Jan 2025.

[24] SPLADE authors & contributors. SPLADE (GitHub repository). `https://github.com/naver/splade`. Accessed: 2024-07-22.

[25] S. Srinivasan. RAG is dead - why enterprises are shifting to agent-based AI architectures. `https://www.techradar.com/pro/rag-is-dead-why-enterprises-are-shifting-to-agent-based-ai-architectures`, 2024.

[26] TREC 2024 RAG Track Organizers. An overview of the TREC 2024 RAG evaluation procedure. `https://trec-rag.github.io/annoucements/evaluation/`. Accessed: 2025-07-25.

[27] TREC 2024 RAG Track Organizers. TREC 2024 RAG track guidelines. `https://trec-rag.github.io/annoucements/2024-track-guidelines/`. Accessed: 2024-07-22.

[28] TREC 2024 RAG Track Organizers. What is TREC RAG? `https://trec-rag.github.io/about/`. Accessed: 2024-07-22.

[29] TREC 2025 NeuCLIR Track Organizers. TREC data: 2024 NeuCLIR track. `https://trec.nist.gov/data/neuclir2024.html`. Accessed: 2025-07-25.

[30] TREC 2025 RAG Track Organizers. TREC 2025 RAG corpus. `https://trec-rag.github.io/annoucements/2025-rag25-corpus/`. Accessed: 2025-07-25.

[31] TREC 2025 RAG Track Organizers. TREC 2025 RAG topics released. `https://trec-rag.github.io/annoucements/2025-topics-released/`. Accessed: 2025-07-25.

[32] TREC 2025 RAGTIME Track Organizers. RAGTIME1 collection. `https://huggingface.co/datasets/trec-ragtime/ragtime1`. Accessed: 2025-07-25.

[33] TREC 2025 RAGTIME Track Organizers. TREC 2025 RAGTIME track guidelines. `https://docs.google.com/document/d/13ttauQKJ9ufvfHyHIPEC6ZJk4PGOOYjwQwM1n4hd4x0`. Accessed: 2025-07-25.

[34] Shivani Upadhyay, Ronak Pradeep, Nandan Thakur, Nick Craswell, and Jimmy Lin. UMBRELA: UMbrela is the (open-source reproduction of the) Bing RELevance Assessor. *arXiv preprint arXiv:2406.06519*, 2024.

[35] Vectara. Open-rag-eval. `https://github.com/vectara/open-rag-eval`, 2024.

[36] Tu Vu, Mohit Iyyer, Xuezhi Wang, Noah Constant, Jerry Wei, Jason Wei, Chris Tar, Yun-Hsuan Sung, Denny Zhou, Quoc Le, et al. FreshLLMs: Refreshing large language models with search engine augmentation. *arXiv preprint arXiv:2310.03214*, 2023.

[37] Yue Wang, John M. Conroy, Neil Molino, Julia Yang, and Mike Green. Laboratory for Analytic Sciences in TREC 2024 retrieval augmented generation track. In *The Thirty-Third Text REtrieval Conference Proceedings (TREC 2024), Gaithersburg, MD, USA, November 15-18, 2024*, volume 1329 of *NIST Special Publication*. National Institute of Standards and Technology (NIST), 2024.

[38] Eugene Yang, Dawn Lawrie, James Mayfield, Douglas W. Oard, and Scott Miller. Translate-distill: Learning cross-language dense retrieval by translation and distillation. In *Proceedings of the 46th European Conference on Information Retrieval (ECIR)*, 2024.

[39] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.