An Iterative Multi-agent RAG System for the TREC 2025 DRAGUN Track

Dake Zhang

University of Waterloo, Canada

Abstract

The main goal of the TREC 2025 DRAGUN Track is to advance research on helping people assess the trustworthiness of online news articles via two tasks: question generation (producing critical questions that readers should consider when evaluating a news article's trustworthiness) and report generation (creating well-sourced reports that provide readers with useful background and context for more informed trustworthiness evaluation). In this paper, we describe our organizer baselines for both tasks, including a starter kit made available to participants at the track's launch. This multi-agent system uses an iterative retrieval-augmented generation pipeline consisting of a query generator, segment retriever, information evaluator, question generator, and report generator. The system is available at: `https://github.com/trec-dragun/2025-starter-kit`.

1 Introduction

As the successor to the TREC 2024 Lateral Reading Track [4], the TREC 2025 DRAGUN (**D**etection, **R**etrieval, and **A**ugmented **G**eneration for **U**nderstanding **N**ews) Track [5] advances research on tools that help readers judge the trustworthiness of online news using Retrieval-Augmented Generation (RAG) techniques. The track used 30 news articles as topics, covering a range of sources and subjects. For each article, there were two tasks:

- **Question Generation**: Creating 10 critical questions that readers should ask to assess the trustworthiness of the news article.
- **Report Generation**: Generating a 250-word well-attributed report to offer readers relevant background and context to aid their assessment. This is a RAG task: statements in the report should be grounded, when appropriate, in the MS MARCO V2.1 Segmented Corpus[1].

To lower the barrier to participation, we released a starter kit with all intermediate artifacts so that participants could swap in their own components (e.g., prompts or retrievers) and quickly build full systems. Our system extends ReadProbe [3] with multiple LLM-driven agents that iteratively generate queries, retrieve text segments, evaluate information, and decide when enough evidence has been gathered to write questions and a report. This workflow mirrors how a careful reader might proceed: propose queries, search, assess, and iterate as needed. Implementation details are given in Section 2. We ran the system with two Large Language Models (LLMs): a closed-source model (`GPT-4.1`[2]) and an open-weight model (`gpt-oss-120b`[3]). Both models were used out of the box, without being trained or fine-tuned on task-specific data.

[1]`https://trec-rag.github.io/annoucements/2025-rag25-corpus/#-ms-marco-v21-segmented-corpus`
[2]`https://openai.com/index/gpt-4-1/`
[3]`https://openai.com/index/introducing-gpt-oss/`

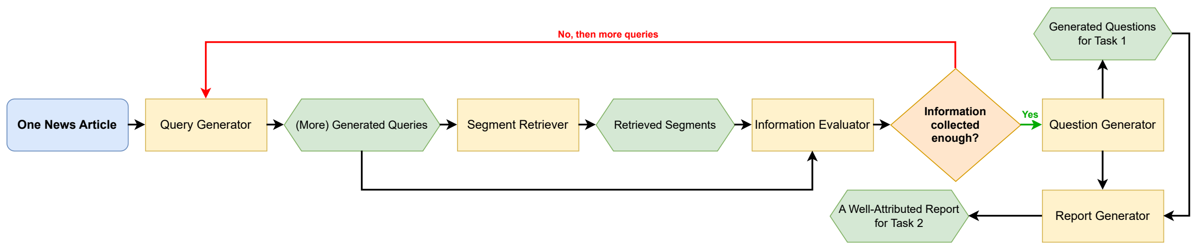

Figure 1: Architecture of the iterative multi-agent starter-kit system. Agents cycle through information gathering and evaluation until deciding that sufficient information has been collected for the final tasks.

For Task 1, we also collected baselines from two commercial systems to estimate the gap between academic systems and leading closed-source applications: ChatGPT 5 Pro Deep Research and Perplexity Deep Research. We provided the same instructions we gave to TREC assessors, with plaintext versions of the news articles via each product's web interface. Unfortunately, these runs are not reproducible: we could not control system prompts, temperature values, or other parameters, and Perplexity did not disclose the underlying LLM for its Deep Research feature. We did not create commercial baselines for Task 2 because reports must be grounded in the specified corpus, whereas the commercial systems use the live web.

Through human evaluation, our starter-kit system achieved comparable performance to ChatGPT 5 Pro Deep Research, second to Perplexity Deep Research. Regarding report generation, our system with `GPT-4.1` achieved significantly higher performance than with `gpt-oss-120b`. Overall, both tasks remain challenging: systems showed limited overlap with expert-created rubrics. In this paper, we share the starter kit and lessons learned to help researchers build better tools that support readers in assessing the trustworthiness of online news.

2 System Implementation

The starter-kit system is designed as a multi-agent iterative pipeline that gathers information from the MS MARCO V2.1 Segmented Corpus and decides when enough evidence has been collected to produce the final outputs. The system consists of several components (agents), each guided by a specialized prompt. Throughout the process, the system maintains a JSON-formatted interaction history of the news article, queries, and retrieved text segments, which later components can consult for context. Figure 1 illustrates the overall architecture of the pipeline. In the remainder of this section, we describe each component in turn, outlining the role it plays in the pipeline.

2.1 Query Generator

The query generator is the first agent in the pipeline. Given an input news article, it generates a set of search queries that a fact-checker might pose. In the first iteration, the system prompt instructs the LLM to produce five detailed queries, each accompanied by a rationale, targeting different facets of the article's trustworthiness (e.g., investigating the source's reputation, verifying key claims, tracing quoted facts, checking alternative viewpoints, etc.). These queries are instructed to be formulated to work well with a traditional keyword-based search engine. If the information gathered later is deemed insufficient by the evaluator (described below), the query generator is invoked again in subsequent iterations to generate additional queries (again in batches of five).

In those later rounds, the agent's prompt is augmented with context about the previous queries and retrieved segments, along with feedback on what information is still lacking. This way, the new queries are informed by what has already been found or missed. The module ensures broad coverage of investigative angles, starting from general and then filling gaps based on the feedback loop from the evaluator.

2.2 Segment Retriever

The segment retriever acts as a search engine to fetch relevant text segments for each query. It implements a three-stage retrieval process to progressively narrow down results and extract the most useful segments from the MS MARCO V2.1 Segmented Corpus, where each segment is a short text span from a web document, with associated metadata.

- **Stage 1: BM25+RM3 Retrieval.** For a given query, we first perform lexical retrieval using Pyserini's implementation of BM25 with RM3 pseudo-relevance feedback [1]. We configure BM25 with parameters $k_1 = 0.9$ and $b = 0.4$, and apply RM3 with 10 feedback terms, 10 feedback documents, and an original query weight of 0.5. This produces an initial list of 1,000 candidate segments from the index.
- **Stage 2: Neural Reranking.** Next, the top candidates from Stage 1 are reranked using a cross-encoder model from the Sentence-Transformers library [2]. Specifically, we use the `ms-marco-MiniLM-L6-v2`[4] cross-encoder, a transformer model fine-tuned on the MS MARCO passage ranking task. For each of the top 1,000 segments, the model examines the query and the segment (title + content) together and produces a relevance score. We then sort the candidates by this neural score and retain the top 20 segments for the final stage. This reranking step adds semantic understanding, helping to prioritize segments that are contextually relevant, even if the wording doesn't exactly match the query.
- **Stage 3: LLM-Based Selection.** In the final retrieval stage, an LLM agent filters the reranked results to pick the most relevant and useful segments. We take the top 20 segments from Stage 2 and present them (along with the original news article and the query) to the LLM with a prompt asking it to select the three most relevant segments for answering the query, ideally from diverse sources. Based on its contextual reasoning, the LLM returns a list of up to three segment IDs (ordered by relevance) that will serve as the evidence for this query. If fewer than three segments are deemed relevant, it selects only those that meet the bar. This LLM-driven filtering helps the pipeline focus on a small set of high-quality pieces of evidence for each query.

The output of the segment retriever for each query is a small set of relevant text segments that feed into the next steps of the pipeline, with the goal of limiting the inclusion of less relevant material that could dilute the context.

2.3 Information Evaluator

The information evaluator is the agent responsible for deciding whether the system has gathered enough evidence to proceed to final output generation. After each batch of queries and their retrieved segments, the information evaluator LLM examines all information collected so far, including the original news article, the list of queries issued, and the content of the top retrieved segments for those queries. It is instructed to be skeptical and to assess the completeness of the

[4]`https://huggingface.co/cross-encoder/ms-marco-MiniLM-L6-v2`

investigation with scrutiny. Essentially, it asks: *Do we now have sufficient information to judge the article's trustworthiness, or should the system dig deeper?*

If the evaluator determines that important questions remain unanswered or key aspects haven't been covered, it will output a negative verdict (that information is not yet sufficient) along with a detailed feedback message explaining what is missing or what lines of inquiry should be pursued next. In response, the pipeline loops back to the query generator, providing it with the feedback and the context of past queries with retrieved segments so it can formulate new queries targeting those gaps. On the other hand, if the evaluator is satisfied that the collected evidence is adequate and no major blind spots remain, it will issue a positive signal indicating that the system can stop searching. This decision triggers the end of the iterative loop. In our implementation, the loop is capped at five iterations to avoid endless cycling and control the budget. Thus, through this component, the system achieves an adaptive, iterative search process, i.e., continuing to retrieve information until a stopping criterion is met.

2.4 Question Generator

Once the information-gathering loop terminates, i.e., the information evaluator deems the collected evidence sufficient, the pipeline moves on to generating the final outputs for the two tasks. The question generator addresses Task 1 (Question Generation) by producing a ranked list of ten critical questions for the given news article. We prompt an LLM with the compiled knowledge base, including the original news article text and all the relevant segments collected during the retrieval iterations, and ask it to formulate the most important questions a reader should consider about the article's trustworthiness. The questions are intended to cover the central issues revealed by the evidence, such as the source's credibility, verification of specific claims, presence of bias or counterpoints, etc., essentially summarizing what a skeptical reader should ask. The question generator does not require additional web searches. It works purely off the information already accumulated, using the LLM's reasoning to synthesize the evidence into question form.

2.5 Report Generator

The final component is the report generator, which produces the report for Task 2 (Report Generation). This stage also uses an LLM prompt to guide the model to act as a professional fact-checker who composes a well-attributed report about the article's trustworthiness. The input to this module includes the news article and the full set of evidence segments retrieved. In addition, we incorporate the questions from Task 1 as guidance. The prompt encourages the LLM to address those important questions within the report. The report generator's output is a concise narrative that provides relevant background and context to help a reader evaluate the article, within the 250-word limit from the track guidelines. Besides, the prompt explicitly instructs the model to include attribution for each claim by citing the IDs of the supporting segments. As per the track requirements, each sentence can cite up to three segment IDs. In essence, the report generator uses the article plus evidence as input and the Task 1 questions as an outline of issues to cover, and it composes a summary that highlights the main trustworthiness factors for the article. This report, together with the question list, constitutes the final output of our baseline system.

2.6 Summary

Overall, the system implementation integrates retrieval-augmented generation in a loop: it generates investigative queries, finds and filters evidence, evaluates what is missing, and only when satisfied does it leverage the collected evidence to produce the reader's questions and report. All intermediate data (queries, selected segments, feedback, etc.) is logged for track participants to use. This approach allows the system to mimic a diligent fact-checker's workflow, i.e., iteratively asking, searching, verifying, and finally reporting. The modular design also makes it easy to swap in improved components, e.g., a better retriever or a different LLM, while preserving the overall pipeline structure. By adapting the prompt that controls each module's behavior, we can make this system generalizable to other similar RAG tasks.

3 Results and Discussion

We ran the starter-kit system with two LLMs: the closed-source `GPT-4.1` and the open-weight model `gpt-oss-120b`. For Task 1 (Question Generation), we also included two commercial "deep research" systems: ChatGPT 5 Pro Deep Research and Perplexity Deep Research, to measure the performance gap between academic systems and commercial products. We provided both products with the same instructions as the TREC assessors and plaintext versions of the news articles through their web interfaces (accessed in late August 2025), but these runs are not reproducible, as the prompts and parameters are proprietary.

During evaluation, TREC assessors created per-topic rubrics consisting of critical questions and expected, short answers, and then evaluated runs against those rubrics. Below, we present results for our organizer runs (both tasks) and compare with the commercial baselines (Task 1 only). The DRAGUN track overview paper [5] covers more details on the evaluation methodology and the results for other submitted runs.

3.1 Task 1: Question Generation

Table 1: Average scores across 30 topics for Task 1 (Question Generation) baselines.

Run ID	**Average Score**
`organizer-t1-perplex`	0.354
`organizer-gpt-oss-t1`	0.235
`organizer-t1-chatgpt`	0.231
`dragun-organizers-starter-kit-task-1` (`GPT-4.1`)	0.226

Task 1 evaluation measures the overlap between each system's generated questions and the assessor's rubric questions. As shown in Table 1, the commercial Perplexity run has the highest average score (0.354), followed by our `gpt-oss-120b` run (0.235), ChatGPT (0.231), and the `GPT-4.1` run (0.226). Using a paired two-tailed Student's t-test, Perplexity is significantly better than the other three runs ($p < 0.01$). We observe no significant differences among runs from ChatGPT, `gpt-oss-120b`, and `GPT-4.1`.

Although Perplexity's average is higher, the per-topic "wins" (counting ties) show notable topic variance: Perplexity leads on 11 topics, ChatGPT on 2, `GPT-4.1` on 1, and `gpt-oss-120b` on 0. No system dominates the majority of topics, showing high variability across articles, consistent with what we observed from the 2024 Lateral Reading Track [4].

Our multi-agent pipeline produces question lists that are statistically comparable to ChatGPT on the rubric-based evaluation, but there remains a clear gap to the top commercial system: Perplexity Deep Research. Given the win distribution and the small margin between `GPT-4.1` and `gpt-oss-120b`, the main bottleneck does not seem to be raw model capability. Promising future improvements could be on better alignment with fact-checkers, e.g., stricter stopping criterion in the evidence-gathering loop.

3.2 Task 2: Report Generation

Table 2: Average supportive and contradictory scores across 30 topics for Task 2 (Report Generation) baselines. Higher supportive and lower contradictory scores are better.

Run ID	**Supportive**	**Contradictory**
`dragun-organizers-starter-kit-task-2` (`GPT-4.1`)	0.230	0.012
`organizer-gpt-oss-t2`	0.150	0.018

For Task 2, the evaluation focuses on answer-level alignment and distinguishes between supportive and contradictory information. Table 2 shows that our `GPT-4.1` run achieves a supportive score of 0.230, significantly better than `gpt-oss-120b` ($p < 0.05$), with a relative gain of about 53%. Contradictory scores are low for both systems (0.012 vs. 0.018) and not significantly different. By per-topic wins (counting ties), `GPT-4.1` leads on 24 topics versus 10 for `gpt-oss-120b`.

The higher supportive score alongside a similarly low contradictory score suggests that `GPT-4.1` synthesizes grounded evidence more effectively without increasing errors. At the same time, the absolute supportive score (0.230) indicates substantial headroom remains, likely in retrieval coverage, evidence selection, and report writing. The specified corpus can also be a limitation, as the rubrics were created using live web data in 2025, while the corpus was collected years before.

When examining the generated reports, we found that the model sometimes included statements such as "*no information was found regarding* [a specific question]" when addressing Task 1 questions. While this behavior aligns with the instruction to respond to each question, it reduces report efficiency under the 250-word limit. These sentences are not rewarded in evaluation and occupy space that could otherwise be used to present relevant supporting information. We will revise the prompt to prefer omission over negative statements unless the absence of evidence is itself informative.

4 Conclusion

This paper presented the organizer baseline system for the question generation and report generation tasks in the TREC 2025 DRAGUN Track. Our system, based on an iterative multi-agent RAG pipeline, simulates the workflow of a professional fact-checker who iteratively asks questions, searches for evidence, evaluates information, and then synthesizes results. In the question generation task, our system performed on par with ChatGPT 5 Pro Deep Research but showed a noticeable gap compared to Perplexity Deep Research. Although our systems with `GPT-4.1` and `gpt-oss-120b` performed similarly in question generation, the former achieved significantly higher performance than the latter in report generation, which suggests that more capable models can benefit the pipeline by better contributing to other components, such as evidence synthesis and report writing. Future work may further enhance the system by assigning different specialized LLMs to corresponding modules, e.g., one optimized for generating investigative queries, another for evaluating retrieved evidence, and another for writing reports. In addition, improving the alignment between report generation and human expectations, especially fact-checkers, may help produce more informative and reader-oriented outputs.

Acknowledgments

This research is funded by Microsoft and the Natural Sciences and Engineering Research Council of Canada (NSERC) grant ALLRP/597573-24.

References

[1] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. Pyserini: A Python Toolkit for Reproducible Information Retrieval Research with Sparse and Dense Representations. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, pages 2356–2362, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380379. doi: 10.1145/3404835.3463238. URL `https://doi.org/10.1145/3404835.3463238`.

[2] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL `https://arxiv.org/abs/1908.10084`.

[3] Dake Zhang and Ronak Pradeep. ReadProbe: A Demo of Retrieval-Enhanced Large Language Models to Support Lateral Reading, 2023. URL `https://arxiv.org/abs/2306.07875`.

[4] Dake Zhang, Mark D. Smucker, and Charles L. A. Clarke. Overview of the TREC 2024 Lateral Reading Track. In *The Thirty-Third Text REtrieval Conference Proceedings (TREC 2024)*, NIST Special Publication. National Institute of Standards and Technology (NIST), 2024. URL `https://trec.nist.gov/pubs/trec33/papers/Overview_lateral.pdf`.

[5] Dake Zhang, Mark D. Smucker, and Charles L. A. Clarke. Overview of the TREC 2025 DRAGUN Track: Detection, Retrieval, and Augmented Generation for Understanding News. In *The Thirty-Fourth Text REtrieval Conference Proceedings (TREC 2025)*, NIST Special Publication. National Institute of Standards and Technology (NIST), 2025.