WaterlooClarke at TREC 2025

Siqing Huo[1] and Charles L. A. Clarke[2]
[1, 2]University of Waterloo, Waterloo, ON, Canada

January 20, 2026

1 Introduction

Participating as the WaterlooClarke group, we focused on the RAG track; we also submitted runs for the DRAGUN Track. For the full retrieval augmented generation (RAG) task, we explored four pipelines: 1) Nuggetizer pipeline. 2) Generate an Answer and support with Retrieved Evidence (GARE). 3) Automatic Retrieval and Generation Plan. 4) Combined. 5) Automatically Selected Best Response. For the DRAGUN task, we explored one-shot prompting with feedback in the loop.

2 Experiment Setup

We employ a two-stage retrieval pipeline for our experiments. We used BM25 [4] as the first stage retriever, and then Mono-T5 and Duo-T5 [2] as the reranker. Then we select the top 15 passages with an UMBRELA [5] score ≥ 2 as the final retrieved output. Retrievers and rerankers are implemented using the pyterrier toolkit [1].

We use GPT-4o-mini as the underlying LLM for our experiments.

3 Nuggetizer Pipeline

Adopted from Ragnarök [3], we submitted one simple nuggetizer run (`WaterlooClarke_nuggetizer`). In this pipeline, we:

1. Retrieve using the original query.
2. Generate a cited answer using the retrieved passages.
3. Validate each sentence against its citations.
4. Importance filtering: evaluate the importance of each nugget as "vital" or "okay".
5. Combine the "vital" nuggets into one final coherent answer.

4 GARE Pipeline

Generate an Answer and support with Retrieved Evidence. We prompt an LLM for an answer and annotate the answer with any supporting evidence we find. We submitted one GARE run (`WaterlooClarke_garag`). In this pipeline, we:

1. Prompt LLM to generate an answer to the original query.
2. Extract a list of atomic factual claims from the answer: split complex statements into minimal checkable units.
3. For each factual claim:
   - Retrieve evidence from external sources / search.
   - Compare evidence to the claim.
   - Validate the claim OR Modify/rewrite the claim based on retrieved passages.
4. Combine the validated claims into a final natural coherent response.

5 Automatic Retrieval and Generation Plan Pipeline

We first prompt the LLM to generate a retrieval and generation plan, the prompt is shown in Appendix A. Each step is either a retrieval step or an answer step. Then, we execute the plan to obtain a final answer. The output of this pipeline is submitted as `WaterlooClarke_auto_plan`.

6 Combined Pipeline

Our Nuggetizer pipeline in Section 3 retrieved passages using only the original query. Our GARE pipeline in Section 4 and Automatic Retrieval and Generation Plan pipeline in Section 5 performed retrieval with a more diverse set of queries. In our combined pipeline, we replace the output of step 1 of the Nuggetizer pipeline with all the unique passages retrieved from the previous three combined pipelines and feed them into the subsequent steps of the Nuggetizer pipeline. The output of this pipeline is submitted as `WaterlooClarke_combined`.

7 Automatically Selected Best Response Pipeline

We have four responses generated from Nuggetizer pipeline (Section 3, GARE pipeline (Section 4), Automatic Retrieval and Generation Plan pipeline (Section 5) and combined pipeline (Section 6) respectively. We conduct pairwise comparisons of the four responses. For each pair of responses A and B, we prompt the LLM twice—once with A presented first and once with B presented first—to mitigate positional bias. The prompt is shown in Appendix B.

Any tie is broken arbitrarily. The winner of each pairwise comparison advances to the next round, and this process continues until a single final winner is selected. The outcome is submitted as `WaterlooClarke_auto_selected`.

8 DRAGUN

For the Question Generation task, we experiment with one-shot prompting and feedback in the loop. We first generate a set of 10 questions with one-shot prompting and then use the LLM evaluator to evaluate the generated questions. The example and evaluation rubric used in the prompts are adapted from the official evaluation guidelines[1]. Lastly, given the questions and feedback generated previously, we ask the LLM to improve the generated questions.

For the report generation task, we use the GARE pipeline in Section 4 to generate an answer to each of the top 10 questions. Then we combine all the factual claims into one coherent final report.

References

[1] Craig Macdonald, Nicola Tonellotto, Sean MacAvaney, and Iadh Ounis. Pyterrier: Declarative experimentation in python from bm25 to dense retrieval. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, page 4526–4533, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384469. doi: 10.1145/3459637.3482013. URL `https://doi.org/10.1145/3459637.3482013`.

[2] Ronak Pradeep, Rodrigo Nogueira, and Jimmy Lin. The expando-mono-duo design pattern for text ranking with pretrained sequence-to-sequence models, 2021. URL `https://arxiv.org/abs/2101.05667`.

[3] 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, 2024. URL `https://arxiv.org/abs/2406.16828`.

[4] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. Okapi at trec-3. In Donna K. Harman, editor, *TREC*, volume 500-225 of *NIST Special Publication*, pages 109–126. National Institute of Standards and Technology (NIST), 1994.

[1] `https://trec-dragun.github.io/TREC%202025%20DRAGUN%20Track%20Assessing%20Instructions.pdf`

[5] Shivani Upadhyay, Ronak Pradeep, Nandan Thakur, Nick Craswell, and Jimmy Lin. Umbrela: Umbrela is the (open-source reproduction of the) bing relevance assessor, 2024. URL https://arxiv.org/abs/2406.06519.

Appendix

A Prompt for Automatic RAG Plan Generation

You are an advanced search assistant. Your task is to generate a structured execution plan for answering a complex natural language question. You are given a user query. Your task is to generate a retrieval plan, which is a list of ordered actions to comprehensively answer the query by searching and reasoning over multiple aspects. You need to output a retrieval plan as a JSON object and nothing else — a list of steps, where each step is either a retrieve or answer operation. These steps collectively retrieve relevant evidence, extract structured variables, generate contextual summaries, and synthesize a final answer:
Each action is one of:
1. retrieve:

```
{
  "type": "retrieve",
  "query": "<BM25-friendly query string>"
  "depends_on": { // optional
      "field_variables": [
        { "name": "<variable_name>", "step": <step_index> }
      ]
  }
}
```

- The overall plan will use multiple retrieve steps to cover different subtopics or entities.
- Phrase the query with high keyword density, synonyms, and minimal stopwords.
- May use earlier variables, e.g., {some_variable["field"]}. Variables can be included as {variable_name} to be replaced by actual values at runtime.
- Queries must be detailed and self-contained when variables are filled.
- If the query depends on earlier variables, must include "depends_on", otherwise "depends_on" should be omitted.
2. answer:

```
{
  "type": "answer",
  "depends_on": {
    "field_variables": [
      { "name": "<variable_name>", "step": <step_index> }
    ],
    "raw_passages": [<retrieve_step_indices>],
    "generated_texts": [<answer_step_indices>]
```

```
    },
    "produces_variables": [ // optional
      {
        "name": "<variable_name>",
        "prompt": "<LLM prompt using {raw_passage_0},
           {generated_text_2}, etc.>",
        "format": [ // must have if produces_variable is included
          { "name": "<field_name>", "description": "<description>" },
          ...
        ]
      }
    ],
    "text_prompt": "<LLM prompt to produce fluent summary using
       available inputs>",
  }
```

- {raw_passage_#} = one passage from the documents retrieved at retrieve step at index #
- {generated_text_#} = textual output from previous answer step at index #, specifically output from using the text_prompt (not produced variables)
- all prompts are natural language prompts that will be used to prompt LLM, which can mention variables in braces corresponding to variables declared or from depended steps. - format cannot be an empty object! It should provide the name and explanation of each field of the variable object.
ALL field that will be referenced in the future must be specified here! Each entry of 'format' must be a dictionary with a 'name' key and a 'description' key. - text_prompt extracts text that are long-form explanations or interpretations, evidence-rich summaries, or fluent and contextual rich text to accompany structured outputs.
- the overall plan will use intermediate answer steps to extract variables (e.g., lists, attributes).
- the last step of the overall plan must be an answer step, and this final answer steps synthesize all relevant information:

```
{
  "type": "answer",
  "depends_on": {
    "field_variables": { ... },
    "raw_passages": [ ... ],
    "generated_texts": [ ... ]
  },
  "text_prompt": "<final LLM prompt for free-form answer>",
}
```

* Must not contain produces_variables.
* Must output a natural, evidence-based, fluent explanation. Must contain "text_prompt".
* Should make use of all the necessary previous variables, texts and passages to generate the final comprehensive response.
NOTE: all prompts generated must access variables' fields in dictionary format (ex: var['field_name']). (Do NOT use dot format)
Example:
Input query: How does urbanization affect bird species diversity across regions?
Output plan:

```
[
  {
    "type": "retrieve",
    "query": "scientific studies on bird species diversity in urban
       environments across different regions"
  },
  {
    "type": "answer",
    "depends_on": {
      "raw_passages": [0]
    },
    "produces_variables": [
      {
        "name": "urban_diversity_records",
        "prompt": "Given {raw_passage_0}, extract a study record
           with: location, study_period, and species_count.",
        "format": [
          { "name": "location", "description": "Geographic region of
             the study (e.g., city or country)" },
          { "name": "study_period", "description": "Time span the
             study covers (e.g., '2010-2015')" },
          { "name": "species_count", "description": "Number of bird
             species observed in the urban area" }
        ],
      }
    ],
    "text_prompt": "Based on {raw_passage_0}, summarize the bird
       species diversity observed in the urban study. Include
       themes, findings, and factors affecting diversity.",
  },
  {
    "type": "retrieve",
    "depends_on": {
```

```
    "field_variables": [
      { "name": "urban_diversity_records", "step": 1 }
    ],
  },
  "query": "bird species diversity in rural areas near
     {urban_diversity_records[\"location\"]} during
     {urban_diversity_records[\"study_period\"]}"
},
{
  "type": "answer",
  "depends_on": {
    "field_variables": [
      { "name": "urban_diversity_records", "step": 1 }
    ],
    "raw_passages": [2]
  },
  "produces_variables": [
    {
      "name": "rural_diversity_records",
      "prompt": "Using {raw_passage_2}, extract rural bird
         diversity records near
         {urban_diversity_records[\"location\"]} and during
         {urban_diversity_records[\"study_period\"]}.",
      "format": [
        { "name": "location", "description": "Rural area near the
           urban location" },
        { "name": "study_period", "description": "Matching time
           period from the urban study" },
        { "name": "species_count", "description": "Number of bird
           species observed in the rural setting" }
      ],
    }
  ],
  "text_prompt": "Summarize findings from {raw_passage_2} on rural
     bird species diversity. Include themes, findings, and
     factors affecting diversity.",
},
{
  "type": "answer",
  "depends_on": {
    "field_variables": [
      { "name": "urban_diversity_records", "step": 1 },
      { "name": "rural_diversity_records", "step": 3 }
    ],
```

```
      "generated_texts": [1, 3]
    },
    "produces_variables": [
      {
        "name": "diversity_comparisons",
        "prompt": "Compare {urban_diversity_records} with
           {rural_diversity_records} for each matching location and
           time period. Extract: urban_species, rural_species,
           difference, and more_diverse_in.",
        "format": [
          { "name": "location", "description": "Study location being
             compared" },
          { "name": "study_period", "description": "Time period of
             the study" },
          { "name": "urban_species", "description": "Species count in
             urban setting" },
          { "name": "rural_species", "description": "Species count in
             rural setting" },
          { "name": "difference", "description": "Difference between
             urban and rural species count" },
          { "name": "more_diverse_in", "description": "Which setting
             had more diversity: 'urban' or 'rural'" }
        ],
      }
    ],
    "text_prompt": "Based on {generated_text_1} and
       {generated_text_3}, explain the relative bird species
       diversity in urban vs. rural environments for each region.",
  },
  {
    "type": "answer",
    "depends_on": {
      "field_variables": [
        { "name": "diversity_comparisons", "step": 4 }
      ],
      "generated_texts": [1, 3, 4]
    },
    "text_prompt": "Using {generated_text_1}, {generated_text_3},
       and {generated_text_4}, write a comprehensive answer to:
       'How does urbanization affect bird species diversity across
       regions?' Highlight patterns, exceptions, and key findings
       across the studies.",
  }
]
```

B Prompt for Pairwise Response Evaluation

Given a user query and two responses (Response A and Response B), evaluate the two responses across the following five dimensions:

- Completeness: Which response more fully answers the query?
- Usefulness: Which response is more helpful or informative in a practical sense?
- Naturalness: Which response is more fluent, coherent, and human-like in tone?
- Conciseness: Which response conveys its information more succinctly without unnecessary detail or repetition?
- Structure: Which response is better organized, with a clear and logical flow of ideas that enhances readability and comprehension?

After evaluating each dimension, provide a final overall preference, indicating which response a typical human would prefer overall. For each dimension and the overall preference, choose one of the following:
- A: if Response A is better
- B: if Response B is better
You must carefully analyze both responses in the context of the query, assess them along specific quality dimensions (completeness, usefulness, naturalness, conciseness, structure), and determine which response performs better in each category. Finally, you must provide a clear overall judgment reflecting which response a typical human would prefer. All evaluations must be based solely on the provided text, with no added assumptions or external information. Returned evaluations for all metrics should be a single character of either 'A' or 'B'.
Search Query: {query}
Response A: {response_A}
Response B: {response_B}
Evaluation: