# Monster Ranking

Charles L. A. Clarke Siqing Huo Negar Arabzadeh

School of Computer Science
University of Waterloo
Canada

## 1 Introduction

Participating as the UWClarke group, we focused on the RAG track; we also submitted runs for the Lateral Reading Track. For the retrieval task (R) of the RAG Track, we attempted what we have come to call "monster ranking". Largely ignoring cost and computational resources, monster ranking attempts to determine the best possible ranked list for a query by whatever means possible, including explicit LLM-based relevance judgments, both pointwise and pairwise. While a monster ranker could never be deployed in a production environment, its output may be valuable for evaluating cheaper and faster rankers. For the full retrieval augmented generation (RAG) task we explored two general approaches, depending on if generation happens first or second: 1) Generate an Answer and support with Retrieved Evidence (GARE). 2) Retrieve And Generate with Evidence (RAGE).

## 2 Monster Ranking

At the beginning of August, we defined our monster ranking process as follows:

1. Expand the initial query $Q$ in as many ways as possible to give queries $Q_0, Q_1, ....$
2. Run each $Q_i$ on as many different ranking stacks as possible to give rankings $R_{ij}$.
3. Pool the top $k$ from each $R_{ij}$ and perform LLM-based graded relevance assessment on the pool. Optionally perform secondary pairwise assessment on the top graded passages from the pool.
4. Fuse the $R_{ij}$ using reciprocal rank fusion (RRF)[4] to give $\mathcal{R}$.
5. Stably sort $\mathcal{R}$ by relevance grade, so that the ordering of $\mathcal{R}$ is retained in each grade. The result is a monster ranking $\mathcal{M}$.

| | Description | runid | |
|---|---|---|---|
| | | $Q_0$ | $Q_1$ |
| $R_{00}, R_{10}$ | BM25F (`title:` = 1; `segment:` = 1 ; `headings:` = 0) | - | `uwcBQ` |
| $R_{01}$ | BM25F (*weights above*) + RSJ PRF | `uwcBA` | |
| $R_{02}, R_{12}$ | Pyserini BM25 + RM3 PRF | - | - |
| $R_{03}, R_{13}$ | DiskVectorIndex using Cohere Embed V3 model | `uwcCQ` | `uwcCQA` |
| $R_{04}, R_{14}$ | Pyserini QL | - | - |
| $R_{05}, R_{15}$ | Reranking top-100 from Cohere Embed V3 by RankGPT | `uwcCQR` | `uwcCQAR` |
| $R_{06}, R_{16}$ | SPLADE-v3 | - | - |
| $R_{07}, R_{17}$ | TCT-ColBERT | - | - |
| | Reciprocal Rank Fusion | `uwc0` | |
| | Monster | `uwc1` | |

Table 1: Summary of rankings ($R_{ij}$) used by Step 2 of the `uwc1` monster ranking. Note that what would be $R_{11}$ was not generated, so that the pool for Step # 3 consisted of 15 runs. The reciprocal rank fusion of these runs was submitted as `uwc0`. Some runs were submitted individually to the track under the runids indicated in the last two columns. BM25F field weights are based on ad-hoc experimentation on TREC Deep Learning test collections, rather than a formal parameter sweep, but we could not find a non-zero value for `headings:` that provided any benefit.

For query expansion (Step #1) we tested various published and idiosyncratic methods through pilot experiments on older TREC Deep Learning test collections. One method in particular gave such a consistent improvement relative to the others that we decided to use only two queries for each topic: $Q_0$, the original query; and $Q_1$, which adds a generated answer to the original query.

For all LLM support we employed GPT-4o as it existed in early August. To generate answers for $Q_1$ we used the system message:

```
You are a helpful assistant with outstanding general knowledge.  Your
user likes questions answered simply and factually using as few words
as possible.
```

Relative to older models, GPT-4o is remarkably good at formatting output according to stated requirements. On average, $Q_1$ is 3.6 times larger than $Q_0$, including the original question. Subjectively, the answers are clear and concise, with very little chit-chat or junk.

Table 1 summarizes the 15 runs contributing to the `uwc1` monster ranking. Thee runs were pooled to depth 20 and judged on a 4-point relevance scale using the prompt of Faggioli et al. [5]. Top-graded qrels were then judged pairwise following the procedure of Clarke et al. [2] but using LLM-based assessments, rather than crowdsourced human judgments. Pairs were judged twice, with a pair A/B also judged as B/A. If the judgments were inconsistent, the tie was broken in favor of the longest passage. The reciprocal rank fusion of these runs ($\mathcal{R}$) was submitted as `uwc0` (Step #4). As a final step (Step #5) `uwc0` was re-ranked

according to the relevance judgments and submitted as `uwc1`. Overall, end-to-end query processing time for this monster ranking process took 12 minutes *per query*, at a cost of $1 per query.

As a second attempt at monster ranking, we started with the official baseline run and judged the top 25 using the prompt from Arabzadeh and Clarke [1], essentially jumping directly to Step #3. We then re-ranked the baseline (Step #2) and submitted it as `uwc2`. Finally, we fused `uwc2` and `uwc2`, submitting it as `monster`.

Figure 1 show pre-conference results, based on differences from the median NDCG@20 scores.

# 3 Retrieval Augmented Retrieval

For the full retrieval augmented generation (RAG) task we explored two general approaches.

**GARE** 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 (`UWCgarag`).

**RAGE** Retrieve-And-Generate-with-Evidence. We retrieve passages and then generate an answer directly based on the passages. We submitted two RAGE runs (`UWCrag` and `UWCrag_stepbystep`).

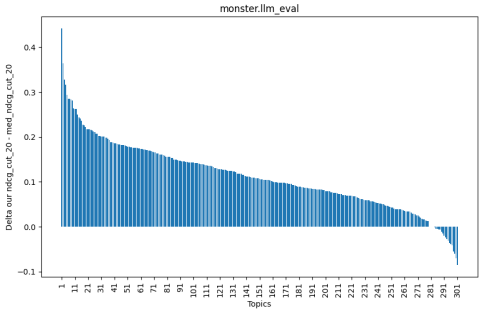
For all submissions, the retrieval stage used `uwcCQA`. We did not use monster ranking for our RAG submissions since we believe it could not be realistically deployed in any production environment, and here we are interested in solving the end-to-end problem.

For `UWCgarag`, our only GARE run, we first prompted an LLM (GPT-4o) to generate a complete answer for the original query $Q_0$, longer and more complete than than the one used for $Q_1$:

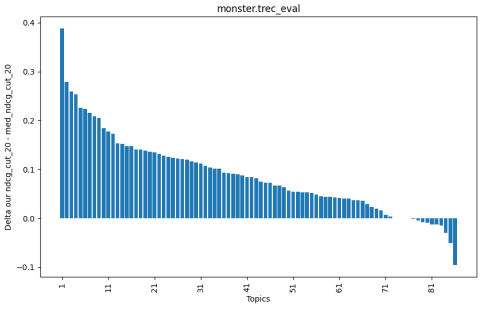
```
You are a helpful assistant with outstanding general knowledge.  Your
user likes questions answered in a concise but complete paragraph, about
a dozen sentences long.
```

Then, we prompt the LLM with the original query $Q_0$, the longer generated answer, and a list of retrieved evidence. The LLM is prompted to attribute each sentence of the generated answer to one or more reference documents. The LLM is also asked to delete sentences that cannot be attributed and modify sentences that are contradicted by the evidence. The overall approach operationalizes Huo et al. [6].

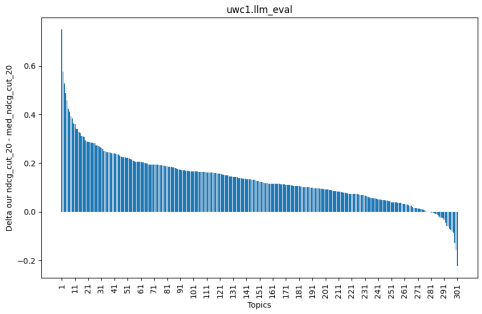
```
Given a query, a proposed answer and a list of reference documents,
attribute each sentence of the proposed answer to one or more reference
documents.  If a claim in the answer cannot be attributed, it should
be omitted from the final output.  If a sentence in the answer is contradicted
by a reference document, it should be modified based on the reference
```



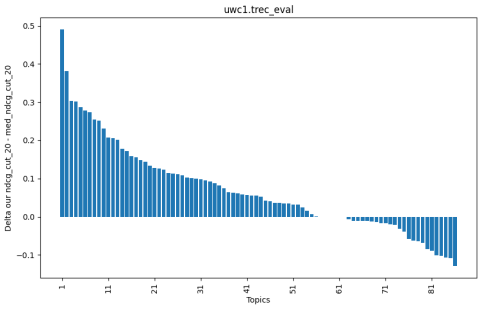

(a) `monster` under LLM Evaluation



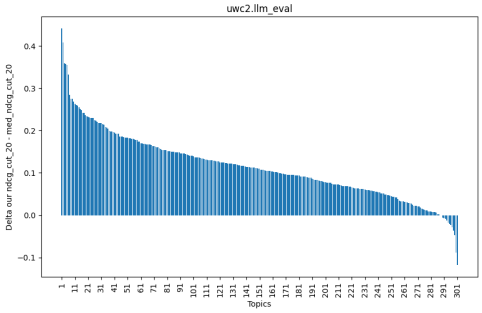

(b) `monster` under Human Evaluation



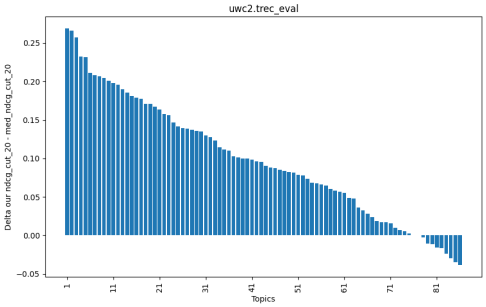

(c) `uwc1` under LLM Evaluation




(d) `uwc1` under Human Evaluation




(e) `uwc2` under LLM Evaluation




(f) `uwc2` under Human Evaluation

Figure 1: Comparison of monster submissions under LLM-based vs. human-based evaluation processes. Each bar shows the difference from the median NDCG@20 for a topic, with topics sorted by decreasing difference. Perhaps because the submissions themselves incorporate an LLM-based evaluation process, performance under LLM-based evaluation appears superior.

```
document and then attributed to it.  Remember the reference documents
are the source of truth and the answer can be modified based on them.
Each sentence in the answer should have its own list of citations, and
citations should be done using the document number (its index in the
list of references).
```

For our RAGE submission `UWCrag` we simply prompted the LLM to generate an answer and to attribute each sentence of the generated answer in one prompt:

```
Given a query and a list of reference documents, generate a summarized
answer, with each sentence attributed to one or more reference documents.
Each sentence in the answer should have its own list of citations, and
citations should be done using the document number (its index in the
list of references).
```

For `UWCrag_stepbystep` we experimented with breaking generation and attribution into more fine-grained steps, where each prompt only handles one simple task. We first prompt the LLM to generate an answer using the evidence:

```
Given a question and a list of reference documents, write an accurate,
engaging, and concise answer for the given question using only the provided
reference documents (some of which might be irrelevant).
```

Then, for each sentence in the generated answer, we prompted the LLM to support it:

```
Given a claim and a list of numbered reference documents, cite the given
claim.  Respond with a list of reference document number that support
the given claim.
```

For all RAG task experiments, we used one-shot prompting where an example is provided in the prompt.

# 4 Lateral Reading

For the Question Generation task, we experimented with one-shot prompting and the system message:

```
Perform lateral reading on web document to evaluate its credibility
by posing 10 questions the reader should ask to evaluate its trustworthiness,
ranked from the most important to the least important to ask.
```

For the Document Retrieval task, we used MonoT5 to rerank all the documents and then using DuoT5 to rerank the top 10 documents.

# 5 Conclusion

We do not argue that the specific process of Section 2 represents the only or best monster ranking process. For example, the results of the pooling process in Step #3 might be used to guide the fusion process in Step #4 by eliminating poorly performing runs from the fusion. Multiple prompts and LLMs might be tried in Step #3, with the results combined. Of course it's always possible to add moooar rankers, with moooar query expansions. We might even loop the whole process, prompting an LLM to extract nuggets from judged relevance passages and generating new expansions from the result. It is harmless to add rankings because the judging process will weed out the non-relevant results. It's monster ranking, not kitchen sink ranking. Other, entirely different, monster ranking processes might work better than ours. The core idea is to throw cost and efficiency out the window in order to produce an ideal ranking, which can be used for evaluation purposes [3].

# References


[1] Negar Arabzadeh and Charles L. A. Clarke. A comparison of methods for evaluating generative ir, 2024. URL `https://arxiv.org/abs/2404.04044`.

[2] Charles L. A. Clarke, Alexandra Vtyurina, and Mark D. Smucker. Assessing top-$k$ preferences. *ACM Transactions on Information Systems*, May' 2021.

[3] Charles L.A. Clarke, Alexandra Vtyurina, and Mark D. Smucker. Offline evaluation without gain. In *ACM SIGIR on International Conference on Theory of Information Retrieval*, page 185–192, 2020.

[4] Gordon V. Cormack, Charles L A Clarke, and Stefan Buettcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 758–759, 2009.

[5] Guglielmo Faggioli, Laura Dietz, Charles L. A. Clarke, Gianluca Demartini, Matthias Hagen, Claudia Hauff, Noriko Kando, Evangelos Kanoulas, Martin Potthast, Benno Stein, and Henning Wachsmuth. Perspectives on large language models for relevance judgment. In *ACM SIGIR International Conference on Theory of Information Retrieval*, page 39–50, 2023.

[6] Siqing Huo, Negar Arabzadeh, and Charles Clarke. Retrieving supporting evidence for generative question answering. In *1st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, page 11–20, 2023.