Intelligent News Comprehension through Query Expansion and LLM-Augmented Generation

Jack Cheverton, Oliwia Majtyka & Ting Liu

Siena University Institute for Artificial Intelligence
515 Loudon Road
Loudonville, NY 12211
jt18chev, o20majt, tliu
@siena.edu

Abstract

This paper discusses our work and participation in the Text Retrieval Conference (TREC) Detection, Retrieval, and Augmented Generation for Understanding News (DRAGUN) track of 2025. In our current digital landscape, it can be difficult to determine the accuracy of what we see online. This is especially true with the rise in fake news. The DRAGUN track challenges participants to develop a system that analyzes an article and generates a list of questions a thoughtful reader should ask if they are trying to determine trustworthiness, as well as generate a report that answers many of these questions. This paper discusses our team's use of query expansion techniques and Large Language Models to approach this task.

1. Introduction

The Detection, Retrieval, and Augmented Generation for Understanding News (DRAGUN) track is a program in the Text Retrieval Conference (TREC), which is funded by the National Institute of Standards and Technology (NIST) and the U.S. Department of Defense and focuses on supporting research in information retrieval and extraction. The DRAGUN track was first run in 2025 and is the continuation of the Lateral Reading track from 2024.

Given the significant rise of fake news in recent times, it can be difficult to evaluate the accuracy of a given news article. This track aims to support readers in assessing the trustworthiness of online news. This is achieved by developing a system that generates questions a thoughtful reader should ask when judging an article's trustworthiness as well as generating a report that provides additional context about that article (Zhang et al. 2025).

To participate in this track, we had a team of two undergraduate researchers and one professor of computer science develop a system to achieve the goals of the track over the course of 6 weeks. This paper discusses our work and participation in the TREC DRAGUN track of 2025.

2. Track Overview

The focus of the DRAGUN track is to develop systems that can help assist readers in assessing the accuracy of an online news article. To achieve this, the track is split into two different tasks:

The first task is question generation. A system should be able to generate ten questions based on a given news article that focuses on questions readers should consider when evaluating the trustworthiness of an article. For example, these questions should ask about the credibility of the author, the reputation of the publisher, any additional context about events described, and more. Each question should avoid being overly general or being a compound question and should not exceed 300 characters in length.

The second task is report generation. This involves creating a detailed report that provides readers with additional information and context about a given news article, allowing readers to better judge the credibility of the article. The questions generated in Task 1 may be used when generating this report. The total word count of the report should not exceed 250 words. When generating the report, the web collection MS MARCO V2.1 (segmented) may be used. This collection contains roughly 114 million text segments gathered from around 11 million online web articles.

For evaluation, all participants receive 30 articles. The associated contextual information is contained within the track's web collection.

3. Literature Review

Previous fact checking techniques have been created to help determine the credibility of a given news article. One of these techniques is known as lateral reading, where a reader "[leaves] a website and [opens] new tabs along the browser's horizontal axis, drawing on the resources of the Internet to learn more about a site and its claims" (Brodsky et al. 2021). This track encourages participants to use lateral reading in their system to determine the accuracy of a given news article by having the system check a large web collection for additional information. Additionally, a study from the University of Regensburg and the University of Santiago de Compostela found that using multiple different search queries and gathering queries from different parts of an article can yield better results when reading laterally (Elsweiler et al. 2025). This suggests that using query expansion techniques when searching for information in the given web collection will improve system performance.

The DRAGUN track is a successor to the 2024 TREC Lateral Reading track. The Lateral Reading track required participants to generate questions based on a given article that a thoughtful reader should ask when evaluating the trustworthiness of a news article. Using these questions, participants would then gather documents that would best help answer these questions. The first task of each track is the exact same, while the second task of each track differs (Zhang et al. 2024).

Out of all runs submitted for Task 1 of the Lateral Reading track, all of them used a Large Language Model (LLM) to generate the 10 questions. A total of 18 runs were submitted, with the top three runs using GPT-4o as their LLM. However, different LLMs appeared in various positions in the rankings, suggesting that the effect of a specific model on performance is much more negligible compared to other techniques (Text Retrieval Conference, 2024).

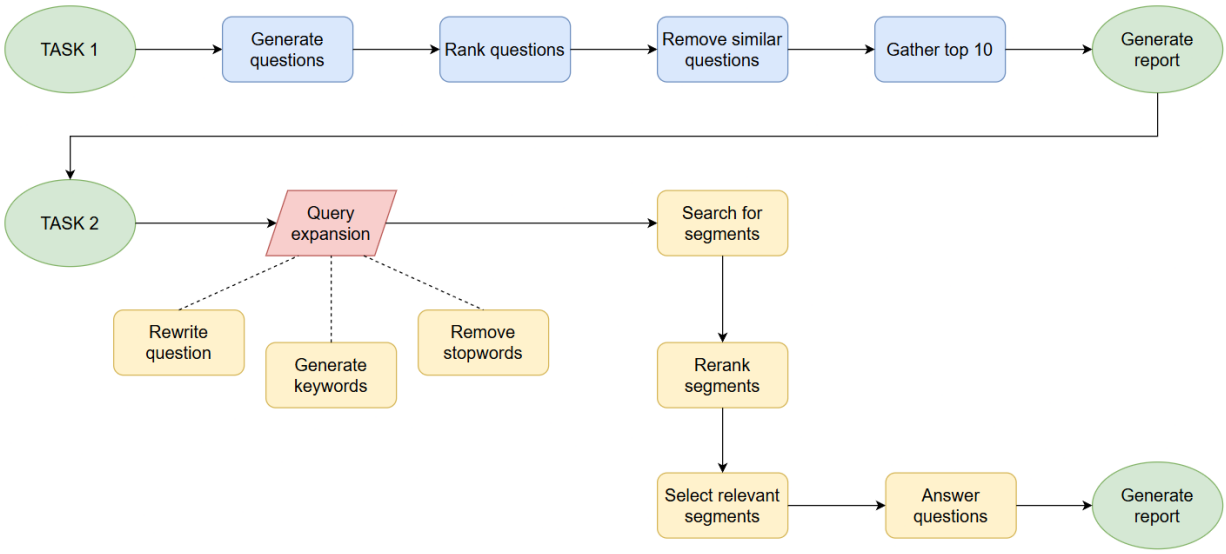

Fig 1: Flowchart of our system

4. System Overview

Our system tries to solve both Task 1 and Task 2. For Task 1, we generate a set of questions using an LLM based on an article, rank the questions based on a pre-trained model, then choose the top 10 questions. For Task 2, each question is answered and the answers are placed into a report for the article. Each question is used in a query expansion process to generate search terms. These terms are used to search through a large text-based dataset for text segments to answer the question. These segments are reranked based on the question, and the top 10 are given to an LLM to answer the question. Finally, each answer is placed in the article report. Figure 1 shows a high-level flowchart of our system.

5. Task 1 Approach

In Task 1, the objective was to generate and select high-quality investigative questions to support the evaluation of a news article's trustworthiness. The process involved three separate calls to an LLM, with each call producing twenty questions, resulting in a total of sixty questions. We decided to call the LLM multiple times because the probabilistic nature of LLMs allows each execution to produce slightly different outputs. This approach increases diversity, ensuring that insightful questions generated in one run are not missed by the others.

All LLM parameters were kept at their default settings, and the prompt was adapted from the DRAGUN starter kit. It instructed the model as follows:

You are a professional fact-checker and media literacy expert. Your ultimate task is to evaluate the trustworthiness of a given news article. Given the text of an article, your task is to generate 20 critical and investigative questions that a thoughtful reader should ask when assessing the article's trustworthiness. These questions should help readers evaluate

aspects such as source bias, motivation, breadth of viewpoints, and overall credibility. These questions should also be able to be answered by people who have never read the original article.

Follow these key principles for question generation:

1. Investigate the Source:
- Generate questions about the publisher's (or key information sources such as organizations, experts, reporters, etc.) background, reputation, and potential biases.
- Ask about their credentials, expertise, and past work.
- Inquire about ownership, funding sources, and editorial policies.

2. Examine Claims and Evidence:
- Question the central factual assertions made in the article.
- Ask about the quality and reliability of evidence presented.
- Inquire about missing context or alternative interpretations.

3. Trace Information Origins:
- Generate questions about the original sources of quotes, statistics, or claims.
- Ask about the methodology behind any data or research cited.
- Inquire about the chain of information from primary sources.

4. Assess Perspective and Balance:
- Ask about missing viewpoints.

5. Evaluate Context and Timing:
- Generate questions about the broader context surrounding the story.
- Ask about the timing of publication and potential motivations.
- Inquire about related events or developments that might provide additional context.

Requirements for each question:
- Maximum 300 characters long.
- Should require information outside of the article (i.e., avoid questions that the article itself can answer), such as search engine results.
- Should only cover one topic.
- Should not be about if a person can be contacted.
- Avoid using terms like "we" and "you".
- Should never be a compound sentence (No single question generated should be able to be split into two or more other questions).
- Should include enough context so that they are understandable without the article. Most importantly, refer to sources
cited in the article by their full name explicitly. Thus, avoid questions such as: "What are the potential biases of the experts cited?" because
without having read the article, one cannot understand who are the "experts cited in the article". Instead, explicitly spell out the names and
positions of the experts cited in the article.

- Be specific to the article's content, not overly general.

Once the sixty questions were generated, a transformer-based regression model was used to score and rank them according to quality. The model predicted scores on a 0–4 scale, where 0 represented low-quality questions and 4 indicated highly critical and well-structured questions. The training data came from the 2024 TREC Lateral Reading track, consisting of TSV files with human-annotated question quality scores. After removing duplicate or incomplete entries, the dataset was split into 80% training and 20% test, with a fixed random seed to ensure reproducibility.

The model was built on Microsoft DeBERTa-v3-small, chosen for its strong sentence-level understanding and ability to capture positional information. To prevent overfitting, a dropout layer (p=0.2) was added, and a linear layer mapped the pooled [CLS] output to a single scalar score. Input texts were tokenized with truncation and padding to a maximum of 128 tokens, and a custom PyTorch Dataset supported batched training for efficiency.

Training was carried out over five epochs using the AdamW optimizer with a learning rate of 2e-5 and a batch size of 8. The Smooth L1 Loss function was chosen for its robustness to noisy labels and suitability for bounded numeric targets. Throughout training, the model showed steadily decreasing loss, indicating stable learning. On the test set, it achieved a Mean Squared Error of approximately 2.0, which is reasonable for a 0–4 scoring range. Applied to 181 questions for the article clueweb22-en0036-31-03154, the model achieved an average absolute difference of 0.63 between predicted and human-assigned scores. For the top fifteen predicted questions, scores were generally within 0.1 points of human annotations.

To further ensure diversity, a cross-encoder similarity model (ms-marco-MiniLM-L6-v2) was used to detect and remove redundant questions. When two questions were too similar, the one with the lower predicted score was eliminated. This process resulted in a final set of ten top-ranked questions per topic, which were then used in the Task 1 report and for subsequent analysis in Task 2.

6. Task 2 Approach

Our approach to generating reports for Task 2 was to answer the questions previously generated for each topic in Task 1. We begin by applying a series of query expansion techniques on each question. These queries will later be used to search through a large dataset for information that can be used to answer these questions. By using these queries, we attempt to retrieve relevant information that could not have been retrieved by simply searching the dataset using the original question. In addition to using the original question as a query, we use three different methods of generating more queries from each question:

The first is rewriting the original question. Using an LLM, we rewrite the original question in five different ways. Unlike other parts of our system, we used a local LLM pre-trained on query expansion (s-emanuilov/query-expansion-Qwen2.5-7B-GGUF, specifically using the quanized version query-expansion.Q4_K_M.gguf). All parameters are set to their default values and the prompt used is as follows: You are going to expand queries. Given a question, your job is to find synonyms or other words or phrases related

to the question. Give five of these expanded queries. Separate each query with a comma. Do not separate them by new lines.

The second method is to use the Python library Yet Another Keyword Extractor (YAKE) to extract keywords from the original question. These keywords are retrieved as n-grams, which are groupings of words that occur next to one another in the original question. We set YAKE to retrieve keywords with n-gram sizes of 6 or less. Each retrieved keyword is also given a score for how relevant that keyword is to the original question, where a higher score means the keyword is less relevant. We discarded all keywords that fell above a score threshold of 0.1.

The final method was to filter out certain words from the original question. This was done using the Python library Natural Language Toolkit (NLTK). We generate two queries using this library: the first is generated by applying a part-of-speech tagger (POS tagger) to the original question. Tokens that are tagged either as "." or "," are removed from the query, essentially removing all punctuation from the query. The second query is generated in a similar way, but the list of tags to remove now includes "POS" and "IN". Additionally, all words that appear in NLTK's English stop word list are removed from this second query. Both queries are also made to only include lowercase characters.

Once each query is gathered, the large dataset MS MARCO V2.1 (segmented) is searched to gather segments to be used to answer each question. Each query is used as a search query to retrieve the top n most relevant segments, where n is set beforehand based on the run being generated. Each segment is divided based on the question the search query was derived from. After the search, each question is given a list of segments (with duplicate segments within each list removed).

These segments are then reranked based on the question using the same cross encoder used in Task 1. The top 20 most relevant segments are kept while the rest are discarded. For each question, GPT-4o is called to select the 3 most relevant segments for the question. The LLM may select less than 3 if there are not enough relevant segments. The temperature is set to 0.1 and all other parameters are set to their default values. This is the system prompt used (which is based on the starter kit):

You are an expert assistant tasked with selecting the most relevant text segments to answer a given question about a news article, which will serve as the context for the retrieval-augmented generation module to answer that question.

You will be provided with:

1. *The news article.*
2. *A question about the news article.*
3. *A list of 20 candidate text segments.*

These segments have been retrieved as potentially relevant and are pre-ranked (from most relevant to least relevant), but this ranking might not be perfect.

Your goal is to identify and return at most 3 segments from the candidates that are most helpful for answering the question. The segments you select should be ordered from most relevant to least relevant. If fewer than 3 segments are relevant, return only those that are. If no segments are relevant, return an empty list. Focus on the direct relevance of the

segment to the question in the context of the provided article. Ideally, the selected segments should come from various sources.

The user prompt is as follows (with values inside brackets representing variables):

Here is the title:
{title}

Here are the headings:
{headings}

Here is the news article:
{article}

Here is the question:
{query}

Here are the 20 candidate segments, ranked by estimated relevance (highest to lowest):
{segments}

Please select the (at most) 3 most relevant segments from the list above to answer the question. Return a comma separated list of the segments ids.

Once this is completed, GPT-4o is called once again for each question to provide an answer based only on the given segments. Each answer is then placed into the topic report and the segments used are placed as citations. Sometimes, the LLM is unable to answer the question based on the provided information and generates a response that states this. When this happens, the system attempts to discard the response and move on to the next question. However, this process is not guaranteed to find all responses that cannot answer the question, and some of these responses end up in the final report. This is a rare occurrence, but it is possible. This LLM uses default values for parameters and uses this system prompt:

Given a question and a list of text segments, your job is to generate a sentence that answers the question using the segments. Make sure that the answer you generate would make sense gramatically without the context of the question. Try to make your answers short. Do not make your answer longer than a sentence. Do not use any outside sources to answer the question, only use the provided text segments.

The user prompt simply gives the question and text segments:

Question: {question}
Segments: {segments}

Answers stop being placed into the topic report either when placing another answer into the response would exceed the word count limit of 250 words or there are no more

questions to answer. Once this happens, the report for that topic has been fully generated and the system starts again with the next topic

7. Task 1 Results and Analysis

For Task 1, we submitted only one run (Team02_Task1). Each set of questions the system generated for each given topic was given a score from 0 to 1, with a higher score indicating higher quality questions. Our run had an average score of 0.304, which performed significantly above the median system's score of 0.183.

Many of the questions generated for each topic would directly ask about the credibility of both the publisher and author of the article. The system would usually place these questions at the top of the list of questions, indicating that these were the most important questions a reader should consider. Other questions would ask for direct evidence about a given claim made in the article. For example, one of the top-rated topics for Task 1 discusses the potential impacts of quantum computers on cryptocurrency. The second question for this topic is "What evidence supports the claim that a 4,000 qubit quantum computer could crack Bitcoin's encryption?" The highest and lowest scoring topics asked these two types of questions.

Due to the similarity of questions between topics, it seems that the main influence on the scores of the questions was the subject of the question rather than how the question itself was framed. Because the pre-trained model had no access to the article itself when selecting the best questions, the decisions it was making were completely independent from the content of the article. It could select questions which would not have been useful to the reader, but which scored high based on previous training data. Therefore, it would be up to the LLM to make sure that the question was asking about something that would be relevant to the reader. All of this implies that the LLM may have more of an impact on the score than the pre-trained model. Based on performance, it seems like LLM was able to correctly identify the best information to be questioned much of the time. However, if the system for Task 1 were to be improved further, it seems like the first place for improvement would be with the LLM.

8. Task 2 Results and Analysis

For Task 2, we submitted a total of three runs. Our first run (Team02_Run01_1000SegmentsExpansion) found 1000 segments gathered per query when searching the dataset. Our second run (Team02_Run02_100SegmentsExpansion) only found 100 segments per query. Our third and final run (Team02_Run02_100SegmentsNoExpansion) also only found 100 segments per query, but disabled all query expansion techniques and only used the questions as queries for the dataset.

Each run was given two scores: a supportive score and a contradictory score. The supportive score represents how relevant the report is to the information the evaluators were expecting to receive in the report. The score ranges from 0 to 1, with a higher score indicating a better performance. The contradictory score represents how much the report contradicts the information expected by the evaluators. This score also ranges from 0 to 1, but a lower score represents a better performance. Like with Task 1, each individual

report in the run was given a score and the entire run was given the average of these scores.

Our three runs were evaluated with average supportive scores of 0.155, 0.158, and 0.139 respectively. All three runs performed above the median run of 0.108. The difference between runs that use query expansion and the run that does not suggests that query expansion made a considerable impact on the performance of the run. This is supported by the fact that Run03 was unable to answer certain questions more so than the first two runs. Additionally, the small difference in performance between Run01 and Run02 suggests that gathering 100 segments per query versus 1000 segments had little impact on performance. The additional 900 segments gathered in Run01 could have been irrelevant to the question being answered.

One influence on these scores that could have had a considerable impact was the probabilistic nature of LLMs. If the system during Run01 and Run02 chose the exact same segments to answer a question, it could have created slightly different answers due to the slightly random aspect of the LLM. This is a likely reason as to why Run02, which was able to obtain only a tenth of the segments Run01 could obtain, was still able to outperform Run01. With this in mind, it is possible that Run01 could have performed exactly the same, if not better, than Run02 had it not been for the randomness of LLMs.

The average contradictory scores of the three runs across all topics was 0.007, 0.013, and 0.007 respectively. Each scored higher than the median score of 0.003, meaning our three runs performed worse than the median. This is most likely due to our system not including any verification for the information we were gathering from the dataset. If the system gathered inaccurate text segments, it would have no way of being able to detect it. Additionally, the random aspect of the LLM could unintentionally use wording that could change the meaning of the answer from what was found in the retrieved text segments. Such incidents would likely be extremely rare and have a very limited impact on the generated sentence.

9. Conclusion

We were able to develop a system that performed above the median submission for both Task 1 and Task 2 of TREC's DRAGUN track. For Task 1, the LLM appeared to have a large impact on performance and was responsible for whether a given set of questions would perform well or not. For Task 2, query expansion seemed to make a great difference in system performance. After a certain point, increasing the amount of information retrieved had a negligible impact on performance, with much of the information gathered in certain runs being unimportant. There are many ways in which the system can be changed that would likely increase performance further. For example, the GPT model we used in the system is now outdated with the release of GPT-5. Switching the LLM to a more recent model would likely increase performance. Additionally, with the importance of query expansion shown in our results, further trying to improve this part of the system could continue to increase performance as well.

Acknowledgements

We would like to thank Siena University's Center for Undergraduate Research and Creative Activity (CURCA) for funding our research. Additionally, we would like to thank Siena University's Computer Science department for funding costs related to our system's use of LLMs.

References

Zhang, D., Smucker, M. D., & Clarke, C. L. A. (2025). *Overview of the TREC 2025 DRAGUN track: Detection, retrieval, and augmented generation for understanding news*. In *Proceedings of the Thirty-Fourth Text Retrieval Conference (TREC 2025)*. National Institute of Standards and Technology.

Brodsky, Jessica & Brooks, Patricia & Scimeca, Donna & Todorova, Ralitsa & Galati, Peter & Batson, Michael & Grosso, Robert & Matthews, Michael & Miller, Victor & Caulfield, Michael. (2021). *Improving college students' fact-checking strategies through lateral reading instruction in a general education civics course.* Cognitive Research Principles and Implications. 6. 1-18. 10.1186/s41235-021-00291-4.

Elsweiler, D., Ateia, S., Bink, M., Donabauer, G., Fernández Pichel, M., Frummet, A., ... & Pascual Presa, N. (2025, July). *Query Smarter, Trust Better? Exploring Search Behaviours for Verifying News Accuracy.* In Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 515-526).

Zhang, D., Smucker, M. D., & Clarke, C. L. A. (2024). *Overview of the TREC 2024 Lateral Reading Track*. National Institute of Standards and Technology. https://trec.nist.gov/pubs/trec33/papers/Overview_lateral.pdf

Text Retrieval Conference. (2024). *Lateral Reading Question generation task Appendix*. https://trec.nist.gov/pubs/trec33/appendices/trec2024-lateral-qgen.html