DayDreamer at CQs-Gen 2025: Generating Critical Questions through Argument Scheme Completion

Wendi Zhou and Ameer Saadat-Yazdi and Nadin Kökciyan
School of Informatics,
University of Edinburgh
{wendi.zhou, ameer.saadat, nadin.kokciyan}@ed.ac.uk

Abstract

Critical questions are essential resources to provoke critical thinking when encountering an argumentative text. We present our system for the Critical Questions Generation (CQs-Gen) Shared Task at ArgMining 2025. Our approach leverages large language models (LLMs) with chain-of-thought prompting to generate critical questions guided by Walton's argumentation schemes. For each input intervention, we conversationally prompt LLMs to instantiate the corresponding argument scheme template to first obtain structured arguments, and then generate relevant critical questions. Following this, we rank all the available critical questions by prompting LLMs to select the top 3 most helpful questions based on the original intervention text. This combination of structured argumentation theory and step-by-step reasoning enables the generation of contextually relevant and diverse critical questions. Our pipeline achieves competitive performance in the final test set, showing its potential to foster critical thinking given argumentative text and detect missing or uninformed claims. .

1 Introduction

In this paper, we present a system description for our contribution to the ArgMining 2025 shared task CQs-Gen (Figueras et al., 2025). Critical questions are an approach to evaluating arguments by providing criteria upon which an argument can be accepted. The argument can be considered acceptable if all the critical questions are satisfactorily answered (Walton and Godden, 2005).

In recent years, there has been an increasing interest in developing systems that can automate this process, aiming to improve the efficiency and reliability of argument evaluation. Our approach leverages advanced natural language processing techniques and machine learning algorithms to generate contextually relevant and diverse critical questions. The system we propose not only identifies key components of an argument but also generates questions that challenge the premises, evidence, and reasoning used in forming conclusions. By doing so, it assists in uncovering potential weaknesses or biases within the argument, thus facilitating more rigorous and comprehensive critical thinking.

Our contribution to the CQs-Gen shared task (Figueras et al., 2025) is rooted in an approach that integrates argumentation theory with a large-scale language model, allowing our system to understand complex argument structures. Our system relies on the identification of argument schemes according to the taxonomy defined by Walton (Walton et al., 2008). Code available at DayDreamer[1].

2 Background

In Walton et al. (2008), the authors develop a comprehensive framework of argument schemes from which critical questions can be derived. An argument scheme is a structured pattern of reasoning associated with a common form of argument. These schemes can be used to analyse and evaluate arguments, particularly in everyday discourse where informal logic is often applied. Not only does this work categorise various types of arguments but it also provides critical questions for each scheme that help in assessing arguments. In their work, 26 Argument Schemes are described with associated critical questions. For example, one common scheme is the *Argument from Expert Opinion* shown in Table 1.

Critical questions are employed to scrutinise and challenge arguments constructed using argument schemes. These questions aim to identify potential weaknesses or gaps in the argument. Each argument scheme has its own set of critical questions. For the *Argument from Expert Opinion*, the critical

[1] https://git.ecdf.ed.ac.uk/s2236454/DayDreamer-CQs-Gen

Proceedings of the 12th Argument Mining Workshop, pages 289–294

Argument from Expert Opinion	
Premise	E is an expert in domain D.
Premise	E asserts that A is true (false).
Conclusion	A may plausibly be accepted (rejected).

Table 1: Scheme for *Argument from Expert Opinion*

questions are shown in Table 2.

Critical Questions (CQs)
CQ1: Is E a credible expert in domain D?
CQ2: Is A consistent with what other experts assert?
CQ3: Is E's assertion based on reliable evidence?
CQ4: Is there any bias or conflict of interest?
CQ5: Is the argument plausible irrespective of expert opinion?

Table 2: Critical Questions associated with the *Argument from Expert Opinion*

These questions guide the evaluator in determining the robustness of the argument by challenging them to assess the credibility of the expert, the quality of the evidence, and any external influences that may affect the truth value of the expert's assertion.

3 Related Work

Several works approach the automatic identification of argument schemes as a multiclass classification problem. Starting from raw text, the goal is to label the text according to the scheme of reasoning being used (Visser et al., 2018; Rigotti and Greco, 2019). Others take this a step further and seek to instantiate the scheme based on the input text (Saadat-Yazdi, 2024; Jo et al., 2021; Ruiz-Dolz et al., 2024). The latter approach considers the problem of scheme identification as a two-step process of scheme classification, followed by instantiation, or a direct sequence-to-sequence translation problem. We combine these two approaches by choosing scheme labels that describe the set of schemes we wish to identify first. However, our goal is to automatically find the exact span of text to which a particular scheme applies, as well as the instantiation of the scheme.

Automatic critical question generation is less studied, with Calvo Figueras and Agerri (2024) being the only work that explicitly undertakes this investigation. Several other works, however, touch upon aspects of automated question generation in broader contexts. Mulla and Gharpure (2023) survey a number of approaches ranging from rule-based to neural approaches for automatic question generation, finding that modelling the task as a sequence-to-sequence learning problem seems to be the most promising direction.

4 Critical Question Generation Pipeline

We now introduce the three main stages within our critical question generation pipeline: *Argument Extraction*, *Critical Question Generation* and *Ranking*. Since our pipeline relies on chain-of-thought prompting with LLMs, the output of each stage would be the input for the next one. This conversational structure is depicted in Figure 1.

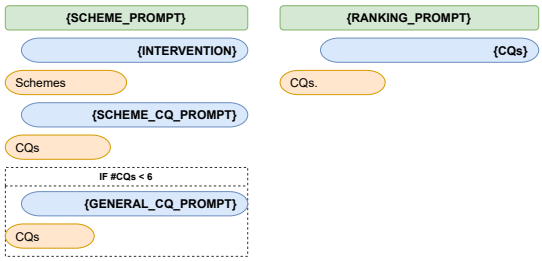

Figure 1: Conversational structure of our approach. The system prompt is shown in green, user prompts in blue, and LLM responses in orange. The text associated with user and system prompts can be found in Appendix A.

Argument Extraction In this stage, we utilised a comprehensive approach to extract arguments with the intervention text as input. Each intervention text was paired with a list of schemes in the provided dataset, which indicates the types of arguments that have been made in the intervention. To utilise this, we collected the definition of all the argument schemes from (Walton et al., 2008) and provided them to LLMs for template instantiation (prompt in Table 4), thereby generating structured arguments. This step provided a structured representation and categorisation of arguments, laying the foundation for critical question generation.

Critical Question Generation After successfully extracting the arguments, the next phase involved generating critical questions pertinent to each scheme. This was also accomplished by referencing Walton's work (Walton et al., 2008), which provides a well-established framework of critical questions for each scheme. With the prompt in Table 5, we complemented the LLMs ability on critical questions generation with this well-defined framework, providing guidance for gener-

ating more relevant and helpful questions by helping the models to hallucinate less. Occasionally, this process would result in fewer than three critical questions. To address this, we introduced one more turn (the dash box in Figure 1) that directly prompts LLMs to generate additional critical questions based on the chatting history when the total number of critical questions is insufficient for the next ranking stage (prompt in Table 7).

Ranking of Critical Questions The final stage of our pipeline focused on ranking the generated critical questions. Ranking is done with a new chat history as we are only interested in the original intervention and the generated critical questions. Using the prompt in Table 6, we present these to LLMs and task them with assessing and ranking the questions based on the helpfulness of the questions. Then, LLMs select the top three most helpful questions as the final output. This ranking process was crucial in choosing the most significant critical questions that would contribute to more in-depth critical thinking, considering the intervention.

5 Results

5.1 Final Evaluation

We obtained the 4^{th} place out of 13 teams that participated, having 60 *Helpful* questions, 25 *Unhelpful* questions and 17 *Invalid* questions. This result comes from our first run result using GPT-4o-mini with manual evaluation.

Figure 2 shows the comparison of our three submissions, where our critical question generation pipeline is combined with two backbone models: *GPT-4o-mini* from OpenAI[2] and *LLaMa-3.1-8B-Instruct* (Grattafiori et al., 2024). Runs 1 and 2 use GPT model twice to assess the stability of our results. Overall, `GPT-4o-mini-run1` achieves the best performance, generating more Helpful critical questions while producing fewer Invalid and Unhelpful ones. `GPT-4o-mini-run2` shows a similar but slightly worse profile, suggesting some instability in our pipeline. In contrast, `LLaMa-7B-run3` demonstrated the lowest response quality compared to other runs, with a tendency toward less helpful and more error-prone outputs. These results highlight the better capability of GPT-4o models in critical question generation compared to LLaMa-7B; however, our pipeline fails to achieve consistent performance in unlocking their full potential.

[2] https://platform.openai.com/docs/models/gpt-4o-mini

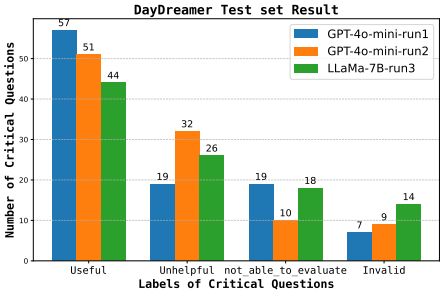

Figure 2: The automated test set evaluation results across three runs. The first two runs are implemented with GPT-4o-mini and the third one is with LLaMa-7B.

5.2 Pipeline Optimization on Validation set

In Table 3, we list all experiment results on the validation set that we conduct to optimize our critical question generation pipeline. Although the baseline method, where we simply prompt the GPT-4o-mini model with the same instruction as (Calvo Figueras and Agerri, 2024), achieves the highest percentage for *Useful* questions, our optimization goal is to minimize the number of *Invalid* and *Unhelpful* critical questions rather than maximize the number of *Helpful* ones. Focusing solely on having a higher number of *Helpful* questions may lead to overfitting, as 75% of the questions in the validation set are generated by LLMs.

We implement our pipeline both with *direct prompting* of the LLMs as well as *conversational prompting*. For direct prompting, we prompt the LLM separately in each stage of our pipeline, which means we take the output of the previous stage and use it together with the instructions of this stage as the input. On the other hand, we prompt LLMs in a conversational manner by keeping a list of chat history messages. In this way, we only provide this stage's instruction and additional helpful information in the prompt because the response of LLMs from the previous stage already exists in the history messages. When comparing the results from *Con* and *Direct Prompting* (in Table 3), we observe a higher percentage of *Useful* critical questions with a similar percentage of *Invalid* and *Unhelpful* ones. Therefore, we build on top of the conversational prompting method to enhance our pipeline.

Each intervention could be related to a long list of scheme names, and we observe that LLMs tend to hallucinate while having more than two scheme templates as input for argument extraction (Section 4). Initially, we feed those schemes into LLMs

Model	Method	Useful	Unhelpful	Invalid	N/A
LLaMa-3.1-8B-Instruct	Baseline	71.68	12.37	3.23	12.72
	$Con_{+ss+rank-er}$	62.21	12.48	2.71	22.60
GPT-4o-mini	Baseline	72.04	13.80	3.94	10.22
	Direct Prompting	56.81	12.19	1.79	29.21
	Con	62.90	13.08	1.25	22.76
	Con_{+ss}	65.41	13.26	3.76	17.56
	$Con_{+ss+rank}$	68.28	12.01	3.94	15.77
	$Con_{+ss+rank-er}$	72.22	8.78	2.87	16.13

Table 3: Validation results to justify our chosen LLMs and the final method of our submitted three run results. All the numbers are the percentage of the number of critical questions with the label. We use **N/A** to represent the fourth label in the automated evaluation: "not_able_to_evaluate". Con is the abbreviation of "Conversational prompting". Con_{+ss} represents that we include "sort scheme" technique on top of the conversational prompting design. Similarly, $Con_{+ss+rank}$ represents that we include prompt tuning for ranking, and $Con_{+ss+rank-er}$ means we remove the scheme templates starting with "ER" as input for LLMs. We choose $Con_{+ss+rank-er}$ as our final submission method for both models we implemented.

with a sliding window where the window size is 2. However, the scheme names within the list are not unique, and the same scheme name could occur in different positions. This window size limits LLMs to extract diverse arguments following the same scheme, as LLMs do not remember what arguments have been extracted with this scheme. To generate more diverse arguments and critical questions, we overcome this challenge with the "sort scheme" technique, where we sort the scheme names in the list and provide all the occurrences of the same scheme names to LLMs together. This approach enables LLMs to estimate the number of argument instances within the intervention that follow the scheme template, thus extracting them all together. There is an evident increase in the number of *Useful* questions and *Invalid* ones from Con to Con_{+ss} in Table 3, justifying that sorting scheme names could result in more diverse critical question generation. Furthermore, we improve the number of *Helpful* questions by modifying the instructions for the ranking stage.

Since our pipeline involves a chain-of-thought prompting, the response of LLMs for each stage could have a great influence on the next stage. We perform a bad case analysis to correlate the quality of the generated critical questions with the scheme types. Unsurprisingly, we notice that most of the *Invalid* critical questions are generated using the schemes that start with "ER" (such as "ERPracticalReasoning", "ERExpertOpinion", etc), which are not defined in (Walton et al., 2008). Since we failed to find the accurate definition, we filled the scheme templates with the corresponding scheme that does not start with "ER". For example, we used the scheme content of "PracticalReasoning" for the scheme "ERPracticalReasoning". However, this inaccurate scheme definition seems to confuse LLMs from extracting correct arguments from the intervention, thus resulting in poor critical question generation. So, we decide not to provide any template to LLMs for these four schemes and let them generate the critical questions based purely on the intervention text. The difference between results from $Con_{ss+rank}$ and $Con_{+ss+rank-er}$ in Table 3 suggests that LLMs can generate higher quality critical questions without misleading scheme templates. Therefore, the quality of the scheme template has a great impact on our pipeline.

6 Conclusion

The findings of our study underscore the significant impact that argument schemes have on the critical question generation process. Our analysis indicates that the accurate definition and implementation of schemes are crucial for extracting valid arguments and enhancing the overall effectiveness of the pipeline. Future work may focus on improving the ability of language models to correctly identify schemes and generate appropriate critical questions accordingly. Constructing a compendium of argument scheme definitions used in the dataset, alongside generating critical questions, would also likely improve results in follow-up work, as it would avoid the issues we found with "ER" schemes.

Limitations

As discussed in our results, the key limitation of this is the lack of definitions of argument schemes for certain cases. We also found that certain schemes used in the dataset were not provided with critical questions in Walton et al. (2008), preventing us from generating critical questions once the scheme has been extracted.

Acknowledgment

This work was supported by the University of Edinburgh-Huawei Joint Lab grants CIENG4721 and CIENG8329.

References

Blanca Calvo Figueras and Rodrigo Agerri. 2024. Critical questions generation: Motivation and challenges. In *Proceedings of the 28th Conference on Computational Natural Language Learning*, pages 105–116, Miami, FL, USA. Association for Computational Linguistics.

Blanca Calvo Figueras, Jaione Bengoetxea, Maite Heredia, Ekaterina Sviridova, Elena Cabrio, Serena Villata, and Rodrigo Agerri. 2025. Overview of the critical questions generation shared task 2025. In *Proceedings of the 12th Workshop on Argument Mining*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Yohan Jo, Seojin Bang, Chris Reed, and Eduard Hovy. 2021. Classifying Argumentative Relations Using Logical Mechanisms and Argumentation Schemes. *Transactions of the Association for Computational Linguistics*, 9:721–739.

Nikahat Mulla and Prachi Gharpure. 2023. Automatic question generation: a review of methodologies, datasets, evaluation metrics, and applications. *Progress in Artificial Intelligence*, 12(1):1–32.

Eddo Rigotti and Sara Greco. 2019. *Inference in Argumentation: A Topics-Based Approach to Argument Schemes*, volume 34 of *Argumentation Library*. Springer International Publishing, Cham.

Ramon Ruiz-Dolz, Joaquin Taverner, John Lawrence, and Chris Reed. 2024. NLAS-multi: A Multilingual Corpus of Automatically Generated Natural Language Argumentation Schemes. *arXiv preprint*. ArXiv:2402.14458 [cs].

Ameer Saadat-Yazdi. 2024. Beyond Recognising Entailment: Formalising Natural Language Inference from an Argumentative Perspective. In *Procedings of the The 62nd Annual Meeting of the Association for Computational Linguistics*.

Jacky Visser, John Lawrence, Jean Wagemans, and Chris Reed. 2018. Revisiting Computational Models of Argument Schemes: Classification, Annotation, Comparison. In *Computational Models of Argument*, pages 313–324. IOS Press.

Douglas Walton and David M Godden. 2005. The nature and status of critical questions in argumentation schemes. In *OSSA Conference Archive*.

Douglas Walton, Chris Reed, and Fabrizio Macagno. 2008. *Argumentation Schemes*. Cambridge University Press, Cambridge.

A Prompts for LLMs

Extract arguments for each of the scheme in {**scheme_name**} from the input paragraph. These schemes are defined as follows: {**scheme_description**} If no argument can be extracted to fit the scheme, extract the main arguments with premise and conclusion.

Table 4: **SCHEME_PROMPT** Prompt for the **Argument Extraction** stage. {**scheme_name**} is the placeholder for the scheme names paired with this intervention. {**scheme_description**} is the placeholder for the scheme definition in (Walton et al., 2008).

{**cq_template**} With the help of the information above, generate a list of critical questions to ask regarding the extracted arguments. You may rephrase the critical question to make it more fluent. Return only a list questions as defined below: [{"CQ1": "the content of the critical question"}, ...]

Table 5: **SCHEME_CQ_PROMPT** Prompt for the **Critical Question Generation** stage. {**cq_template**} is the placeholder for the defined critical question template related to each scheme.

{**intervention**}

A helpful critical question can potentially challenge one of the arguments in the text.
Rank and select top three most helpful critical questions.
Return ONLY the question id in a Python list:
“‘python
[id_1, ...]

Table 6: **GENERAL_CQ_PROMPT** Prompt for the **Ranking of Critical Questions** stage. {**intervention**} is the placeholder for the original intervention text.

{**intervention**}

A helpful critical question can potentially challenge one of the arguments in the text.
Provide me 3 more critical questions that should be asked given the arguments from the text above.
Return only the questions as following format:
[{"CQ1": "the content of the critical question"}...]

Table 7: **RANKING_PROMPT** Prompt for generating more critical questions when the available critical questions are insufficient for ranking.