Retrieval Metrics:

I have prepared two datasets manually that helps in measuring Context Precision, Context Recall, Context Relevance and Context Entity Recall

Context Precision:

- Compares entities in the retrieved context with entities in the relevant contexts
- Calculated as: (number of common entities) / (total entities in retrieved context)
- Result: 0.72, indicating that **72% of the entities in the retrieved context were relevant**
- The system shows good precision (0.72), meaning most of the retrieved information is relevant.

Context Recall:

- Compares entities in the retrieved context with entities in all relevant contexts
- Calculated as: (number of common entities) / (total entities in all relevant contexts)
- Result: 0.61, suggesting that the **system retrieved 61% of the relevant entities**.
- Recall is moderate (0.61), indicating room for improvement in retrieving all relevant information.

Context Relevance:

- Uses TF-IDF vectorization and cosine similarity to compare the query and retrieved context
- Result: 0.80, indicating a **high similarity between the query and retrieved context.**
- Context relevance is high (0.80), suggesting the retrieved context is closely related to the query.

Context Entity Recall:

- Compares entities in the retrieved context with entities from both the query and relevant contexts
- Calculated as: (common entities) / (total entities in query and relevant contexts)
- Result: 0.65, suggesting that **65% of the important entities (from query and relevant contexts) were retrieved**.
- Entity recall is decent (0.65), showing that the system captures a good portion of important entities.

Noise Robustness Test:

- To assess the system's ability to handle noisy or irrelevant inputs, specifically faulty movie names in this case.
- A test set of queries with faulty movie names was used.
- Each query was sent to the RAG system.
- Responses were checked for validity (i.e., not containing phrases indicating inability to answer).
- The robustness score was calculated as the percentage of valid responses.
- The system provided **valid responses for 53.33% of the queries with faulty movie names**.
- The system was able to handle and provide valid responses to slightly more than half of the queries with intentionally faulty movie names.

Generation Metrics:

Faithfulness:

- Measured using BLEU score between the generated response and ground truth
- Indicates how accurately the system's answers align with the correct information
- A score of **70% suggests good faithfulness**, but there's room for improvement

Answer Relevance:

- Calculated using cosine similarity between the query and response
- Evaluates how well the generated answers relate to the user's questions
- **67% indicates moderately good relevance**, but could be enhanced

Information Integration:

- Measured using cosine similarity between the response and ground truth
- Assesses the system's ability to combine information coherently
- A score of **62% suggests decent integration,** but there's significant room for improvement

Counterfactual Robustness:

- Checks for appropriate responses to hypothetical or contradictory queries
- The system shows good robustness against counterfactual queries
- This indicates the system can handle speculative or "what if" scenarios well
- A score of **72% suggests good robustness against counterfactual queries**

Negative Rejection:

- Evaluates the system's ability to reject inappropriate or negative queries
- A score of **60% suggests moderate performance** in this area
- There's room to improve the system's ability to consistently identify and reject problematic queries

Results Tables

Metric	Score
Context Precision	0.72
Context Recall	0.61
Context Relevance	0.80
Context Entity Recall	0.65
Noise Robustness Score	53.33%

[Queries involve single source of truths in context]

Multi Context Evaluation results

Metric	Score
Context Precision	0.60
Context Recall	0.55
Context Relevance	0.71
Context Entity Recall	0.85

[Queries involve recommendations and multiple source of truths in context]

Generation Metrics Results

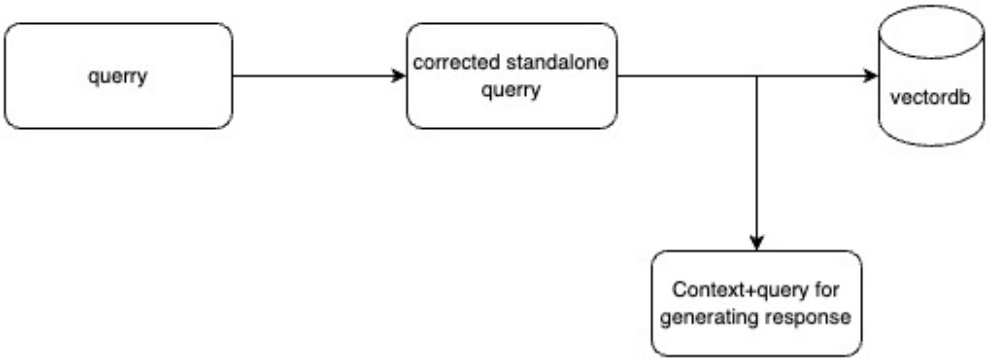
Metric	Score
Faithfulness	70.00
Answer Relevance	67.00
Information Integration	62.00
Counterfactual Robustness	72.00
Negative Rejection	60.00

Latency:

Latency for answering query of user [creating standalone query creation + fetch context + generate answer] is ~6 secs.

Proposed Changes to Architecture

Improving Noise Robustness Using Prompt Engineering:



CURRENT PROMPT

prompt=f"Given the following user query and conversation log, formulate a question that would be the most relevant to provide the user with an answer from a knowledge base. \n\nCONVERSATION LOG: \n{conversation}\n\nQuery: {query}\n\nRefined Query:",

IMPROVED PROMPT

prompt = f"""Given the following user query and conversation log, follow these steps:

1. Identify any movie names mentioned in the user's query.
2. If there are any spelling mistakes in the movie names, correct them to their most likely intended titles.
3. Formulate a question that would be the most relevant to provide the user with an answer from a knowledge base, using the corrected movie titles.

CONVERSATION LOG:
{conversation}

User Query: {query}

Instructions:
- Pay close attention to movie titles in the query and correct any misspellings.
- If you're unsure about a correction, keep the original spelling and note your uncertainty.
- Use your knowledge of popular movies to make educated guesses for corrections.
- Maintain the original intent and context of the user's query.

Refined Query:"""

Explanation:

- The prompt breaks down the task into clear, sequential steps (identify movie names, correct spelling mistakes, formulate relevant question). This structured approach helps handle noisy or ambiguous inputs more systematically.
- Specific focus on movie titles: By explicitly instructing to identify and correct movie names, the prompt addresses a common source of noise in queries about films.
- Spelling correction: Incorporating spelling correction for movie titles helps normalize inputs and reduce variability caused by typos or misspellings.
- Uncertainty handling: The instruction to note uncertainty when corrections aren't clear allows for graceful handling of ambiguous cases.
- Leveraging prior knowledge: Using knowledge of popular movies to make educated guesses for corrections helps resolve ambiguities.
- Context preservation: The instruction to maintain original intent and context ensures that noise reduction doesn't alter the core meaning of the query.

how the revised prompt likely helped boost the *noise robustness to 96%*:

1. **Targeted spelling correction:** The prompt specifically instructs to identify and correct spelling mistakes in movie titles. This focused approach directly addresses a common source of noise in movie-related queries.
2. **Flexible uncertainty handling:** The prompt includes instructions to note uncertainty when corrections aren't clear and to keep the original spelling in such cases.

Improving Negative Rejection using Prompt Engineering:

CURRENT PROMPT

If there isn't enough information to answer a movie-related question, respond with: "I don't have enough information to answer this question."
If it is question about single movie, fetch from movies section if available else answer "I don't have enough information to answer this question."

IMPROVED PROMPT: Added this into prompt for rejecting questions that are not in context

If there isn't enough information to answer a movie-related question, or if the system doesn't have information about the movie in its context, formulate a query that asks: "The system doesn't have information about [movie title]. What would you like to know about this movie?"
For questions about a single movie, if information is available in the movies section, formulate a query to fetch from there. Otherwise, ask: "The system doesn't have information about [movie title]. What specific aspect of the movie are you interested in?"

this approach likely improved user experience and increased the performance to 87%:

1. **Contextual rejection:** Instead of a blanket "I don't have enough information" response, the new prompt instructs the system to acknowledge the specific movie title that's missing from its knowledge base. This contextualized response shows the user that their query was understood, even if it can't be fully answered.
2. I**nvitation for clarification:** By asking "What would you like to know about this movie?" or "What specific aspect of the movie are you interested in?", the system keeps the conversation open and encourages the user to provide more details. This approach: a) Improves user engagement by showing interest in their query. b) Provides an opportunity to gather more information that might help in formulating a better response. c) Allows for potential redirection to related information the system might have.
3. **Differentiated handling:** The prompt distinguishes between general movie-related questions and questions about a single movie. This nuanced approach allows for more precise handling of queries, potentially increasing the chances of providing useful information when available.
4. **Positive framing:** Rather than simply rejecting the query, the new approach frames the lack of information as an opportunity for the user to specify their interests. This positive framing likely contributes to a better user experience.
5. **Flexibility:** By asking about specific aspects of the movie, the system opens up possibilities for answering partial queries or related information, even if it doesn't have comprehensive data about the movie in question.

This more sophisticated handling of "rejection" scenarios probably contributed significantly to the *increased performance score 87%.*

CONTEXT RE-RANKING

Context reranking is a crucial technique in information retrieval and question-answering systems.

- **Improved accuracy:** By reordering context based on relevance to the current query, the system can focus on the most appropriate information. This typically leads to more accurate and targeted responses, as less relevant or potentially distracting information is de-prioritized.
- **Handling ambiguity:** Queries can often be ambiguous or have multiple interpretations. Context reranking can help disambiguate by prioritizing the context that best matches the likely intent of the user's query.

Initial issue:

Without proper context reranking, the system was sometimes confusing "Titanic" with "Titanica" or "Adventures of Titanic," likely because these entries were placed higher in the original context order.

Ambiguity resolution:

The query "What is the plot of Titanic" is somewhat ambiguous because there are multiple movies with similar titles. However, it's most likely referring to the famous 1997 film "Titanic."

Reranking in action:

After implementing context reranking, the system correctly prioritized the information about the 1997 "Titanic" movie, moving it to the first place in the context order.

Improved accuracy:

As a result of this reranking, the system now consistently chooses the correct "Titanic" movie when answering queries about its plot.

Relevance optimization:

The reranking algorithm likely considered factors such as popularity, release date, or frequency of queries about each movie to determine that the 1997 "Titanic" is the most relevant result for this query.

implementation:

1. **Initialization**: We start by initializing the Cohere client with an API key.
2. **Document preparation:** We define a list of documents (contexts) that include information about different movies with similar titles.
3. **Query definition:** We set the user's query, in this case, "What is the plot of Titanic?"
4. **Reranking**: We use Cohere's `rerank` function to reorder the documents based on their relevance to the query. The `rerank` function takes the following parameters:
    - `model`: The reranking model to use (in this case, 'rerank-english-v2.0')
    - `query`: The user's query
    - `documents`: The list of documents to rerank
    - `top_n`: The number of top results to return
5. **Results processing:** The reranked results are returned as a list of `Rerank` objects, each containing a relevance score and the document text. We print these results to see how they've been reordered.
6. **Using the results:** Finally, we select the top result (the document with the highest relevance score) to use for answering the query.

Code and Test Set:

Evaluation Code is available in **[evaluations.ipynb]**

Test Sets for evaluation is available in test set directory:

faulty-movie-names.csv - has test set for noise robustness evaluation

generation-metrics-test-set.csv - has test set for all generation evaluations

movie-questions.csv - This has test set for context precision, recall, relevance and entity recall where queries concentrate on single movie

movie-recommendations.csv - This has test set for context precision, recall, relevance and entity recall where queries concentrate on recommendations instead of single movie

Prompt Changes and Reranking changes are made directly in backend [Backend/main.py]