Query Rewriting Prompts:

Prompt for the Base and Filtered Standalone queries generation and Release Number extraction:

<|begin_of_text|><|start_header_id|>system<|end_header_id|>

You will be given a user question and conversation history. Follow these steps to rewrite the question:

1. Check if the conversation history is empty.
    - If empty, return the user question as standalone_question without modification.
    - If not empty, decide if the user question is a follow-up based on the conversation history.
        - If not a follow-up, return the user question as standalone_question without modification.
        - If a follow-up, then rewrite the user question as a standalone question. Do not change any keyword, names, numbers or abrivations.
2. Ensure the standalone question is clear and precise, keeping the original meaning, order of words, and key terms (like URLs, proper nouns, product names, etc.).
3. Extract the target release name and version number from the standalone question and return it as version. If there is no target release name and version number, return "None".
4. Create a better search query for a web search engine based on the standalone question and return it as better_query. Ensure you do not change any keyword, names, numbers, verbs, or abbreviations.
5. Ensure the better_query is clear, precise, and retains the original meaning and important terms of the standalone question.
6. Your output must be a JSON object only. Do not include any explanation or any other text. You must strictly follow this JSON schema:

{{
    "standalone_question": "*Step 1 result*",
    "version": "*Step 3 result*",
    "better_query": "*Step 4 result*",
}}

Example 1:

Example Question 1:

"What Circuit Packs support it?"

Example Conversation History 1:

"Human: What is iPhone 14 Pro 256GB in 6542 iOS 16?

AI: some answer"

Example Response 1:

{{
    "standalone_question": "What Circuit Packs support the iPhone 14 Pro 256GB in 6542 iOS 16?",
    "version": "6542 iOS 16",
    "better_query": "iPhone 14 Pro 256GB supported Circuit Packs in 6542 iOS 16"
}}

Example 2:

Example Question 2:

"What are the supported downgrade routes for 5832 OTA iOS 15.60?"

```
Example Conversation History 2:
""

Example Response 2:
{{
    "standalone_question": "What are the supported downgrade routes for 5832 OTA iOS 15.60?",
    "version": "5832 OTA iOS 15.60",
    "better_query": "supported downgrade routes for 5832 OTA iOS 15.60"
}}

Example 3:
Example Question 3:
"When I upgrade from 11.8 to 13.50 what happens with MBA-13 w/3xNEC?"

Example Conversation History 3:
""

Example Response 3:
{{
    "standalone_question": "When I upgrade from 11.8 to 13.50 what happens with MBA-13 w/3xNEC?",
    "version": "13.50",
    "better_query": "MBA-13 w/3xNEC upgrade from 11.8 to 13.50 impact"
}}

Example 4:
Example Question 4:
"Describe Apple M1 chip?"

Example Conversation History 4:
""

Example Response 4:
{{
    "standalone_question": "Describe Apple M1 chip?",
    "version": "None",
    "better_query": "Apple M1 chip defenition"
}}

Example 5:
Example Question 5:
"List operational considerations in iPhone restarts"

Example Conversation History 5:
""
```

Example Response 5:
{{
    "standalone_question": "List operational considerations in iPhone restarts",
    "version": "None",
    "better_query": "operational considerations in iPhone restarts"
}}

Example 6:
Example Question 6:
"Is HD/MI display port supported in 15-gen M1-series Macbook?"

Example Conversation History 6:
""

Example Response 6:
{{
    "standalone_question": "Is HD/MI display port supported in 15-gen M1-series Macbook?",
    "version": "15-gen M1-series Macbook",
    "better_query": "HD/MI display port support in 15-gen M1-series Macbook"
}}

<|eot_id|>

<|start_header_id|>Conversation History<|end_header_id|>
{chat_history}
<|eot_id|>

<|start_header_id|>User Question<|end_header_id|>
{question}
<|eot_id|>
<|start_header_id|>Assistant<|end_header_id|>

Prompt for Versionless Standalone query generation:

<|begin_of_text|><|start_header_id|>system<|end_header_id|>

You are an expert at finding a target in a string and removing the target from the string. You are provided with a string called query and a target. Remove the target and any preposition directly preceding it from the query.

1. Analyze the query and find the target. If the target is 'None', return the query as both the answer and complete_answer without modification.

2. Ensure the answer does not contain the target and that it is a complete query.

3. Analyze the answer and check if it has any unnecessary prepositions or extra words:

    - If the answer is complete and does not have any unnecessary prepositions or ambiguity, return the answer without modification as both answer and complete_answer.

    - If the answer has any unnecessary prepositions like 'for,' 'in,' 'to,' etc., remove only those words and return the modified version as complete_answer. Do not remove any product numbers, important keywords, and names that can be useful in a search. Do not remove any word that changes the meaning of the query.

    - Ensure that the complete_answer includes all relevant parts of the answer unless a word is an unnecessary preposition or causes ambiguity.

4. Ensure the complete_answer does not contain the target, is not ambiguous, is a complete query, and contains all of the keywords and product names and numbers as the answer.

5. Your output must be a JSON object only. Do not include any explanation or any other text. You must strictly follow this JSON schema:

{{

    "answer": "*Step 1 result*",

    "complete_answer" : "*Step 3 result*"

}}

Example 1:

Example Query 1:

"Features of the model in Release 20.50"

Example Target 1:

"Release 20.50"

Example Response 1:

{{

  "answer": "Features of the model in",

  "complete_answer": "Features of the model"

}}

Example 2:

Example Query 2:

"Steps for updating from Version A to Version B"

Example Target 2:

"Version B"

Example Response 2:

{{

  "answer": "Steps for updating from Version A to",

  "complete_answer": "Steps for updating from Version A"

}}

Example 3:
Example Query 3:
"Describe port functionality without connection to line port in Module Y for Z Digital Release X"
Example Target 3:
"Z Digital Release X"
Example Response 3:
{{
  "answer": "Describe port functionality without connection to line port in Module Y for",
  "complete_answer": "Describe port functionality without connection to line port in Module Y"
}}

Example 4:
Example Query 4:
"Review specific considerations 234 Version 1.0 pluggable restarts"
Example Target 4:
"234 Version 1.0"
Example Response 4:
{{
"answer": "Review specific considerations pluggable restarts",
"complete_answer: "Review specific considerations pluggable restarts"
}}
<|eot_id|>
<|start_header_id|>Query<|end_header_id|>
{query}
<|eot_id|>
<|start_header_id|>Target<|end_header_id|>
{target}
<|eot_id|>
<|start_header_id|>Assistant<|end_header_id|>

Prompt for Release Number mapping:

<|begin_of_text|><|start_header_id|>system<|end_header_id|>

You are provided with a target and a list of versions. Your task is to find which of the versions matches the target and return the index of that version.

Your output must be a JSON object only. Do not include any explanation or any other text. You must strictly follow this JSON schema:

{{

"index": "*index*"

}}

Examples:

Example 1

List of Versions 1:

"1. iOS 14

2. 2430 release 12

3. None

"

Target 1:

"release 12"

Example output 1:

{{

  "index": "2"

}}

Example 2

List of Versions 2:

"1. iOS 14

2. 2430 release 12

3. None

"

Target 2:

"543"

Example output 2:

{{

  "index": "3"

}}

<|eot_id|>

<|start_header_id|>List of Versions<|end_header_id|>

{list_of_versions}

<|eot_id|>

<|start_header_id|>Target<|end_header_id|>

{target}

<|eot_id|>

<|start_header_id|>Assistant<|end_header_id|>

{{

Prompt for Context Reduction:

<|begin_of_text|><|start_header_id|>system<|end_header_id|>

You are provided with a context and a user question. Your job is to extract all parts of the context that directly help with answering the question and ensuring the answer is correct. Consider synonyms and abbreviations when extracting relevant text. Additionally, include any text that helps understand why the extracted text answers the question, such as headings or introductory sentences. Ensure the text from the context is not modified. Return an empty list [] if the question cannot be answered from the provided context. extract all of the headers and titles. Your output must be a JSON object only following the below schema:

```
{{
  "extracted_text": ["*extracted_text*","*extracted_text*"]
}}
```

Example:

Example context:

"New introduced apple products

– MA 2024

– Ipad Air

– Iphone 10

– Iphone 11"

Example question:

"What new Apple iphones have been introduced?"

Example output:

```
{{
  "extracted_text": [
    "New introduced apple products",
    "– Iphone 10",
    "– Iphone 11"
  ]
}}
```

Example question:

"what new MacbookAir was introduced?"

Example output:

```
{{
  "extracted_text": [
    "New introduced apple products",
    "– MA 2024"
  ]
}}
```

<|start_header_id|>Context<|end_header_id|>

{context}

<|eot_id|>

<|start_header_id|>User<|end_header_id|>

{question}

<|eot_id|>

<|start_header_id|>Assistant<|end_header_id|>

{{

Prompt for Context Selection:

<|begin_of_text|><|start_header_id|>system<|end_header_id|>

You will be provided with a question and a list of contexts.

Your task is to return the ids of the contexts for answering the question ordered from most to least useful.

Ensure you do not miss any id of any contexts.

Your output must be a JSON object only. Do not include any explanation. Use this JSON schema:

{{

  "ids": ["*id*"]

}}

Example:

Example version:

"ios 14"

example question:

"add application to ios 14"

example contexts:

context 1: "to delete an app hold an icon down and click the x icon"

context 2: "to add an app go to app store"

context 3: "ios 14 is great and allows you to install apps"

example output:

{{

"ids": ['2','3','1']

}}

<|eot_id|>

<|start_header_id|>contexts<|end_header_id|>

{contexts}

<|eot_id|>

<|start_header_id|>User<|end_header_id|>

{question}

<|eot_id|>

<|start_header_id|>Assistant<|end_header_id|>

{{

Prompt for Answer Generation:

<|begin_of_text|><|start_header_id|>system<|end_header_id|>

You are a technical chatbot. Your role is to assist customers with their technical queries based on the given context, which includes multiple potential answers. All the potential answers are from the correct version base on the question. Analyze the context, choose the most relevant and detailed chunk, and generate a concise and coherent response based on that. If you can't determine the correct answer, return 'I don't know'.

<|eot_id|>

<|start_header_id|>Context<|end_header_id|>

{context}

<|eot_id|>

<|start_header_id|>User<|end_header_id|>

{question}

<|eot_id|>

<|start_header_id|>Assistant<|end_header_id|>

Base on the context the answer to the question is as follows:\n

Prompts for G-eval based Answer Correctness metric:

1. Compare the 'actual output' directly with the 'expected output' to verify factual accuracy
2. Any mismatch in units or precision values is NOT acceptable and must result in score of zero
3. Assess if there are any discrepancies in details, values, or information between the actual and expected outputs
4. It is OK if the 'actual output' omits some details from 'expected output' as long as the 'actual output' answers the input
5. It is OK for the 'actual output' to include extra information or present details in a different order than the 'expected output'
6. Missing examples are acceptable as long as the main concepts between the 'expected output' and 'actual output' are the same
7. If the 'expected output' is not empty and does not explicitly state that the input cannot be answered, then 'actual output' of 'I don't know' is NOT acceptable and must result in a score of zero
8. If the 'expected output' is empty or explicitly states that the input cannot be answered, then the only acceptable 'actual output' is 'I don't know,' and it should receive full credit