Development of persona-tailored RAG system

8 Dec 2024

J. Spencer Morris
University of California, Berkeley
jspencermorris@berkeley.edu

Abstract

A persona-tailored retrieval augment generation (RAG) proof of concept (POC) system was developed and evaluated using a set of 4 performance metrics inspired by RAGAS but also including, for example, a pairwise metric scored by a judge model. Chunk size and top_k had significant impact on model performance, as did differences in all prompt templates. A series of tests was performed to ablate components of the main RAG prompt template, for which inclusion of persona-specific passage length requirements was associated with improved pairwise score. The three best models had comprehensive judge scores of ∼ 3.7, showing marked improvement over he initial baseline of ∼ 3.0. These findings underscore the potential of persona-tailored RAG for an improved question-answering experience.

1 Introduction

Retrieval-augmented generation (RAG) is a powerful methodology that reduces the hallucination potential of generative language models by incorporating a document store into the QA pipeline.[1] By leveraging retrieved document chunks relevant to the input question, RAG systems can enrich the context for more accurate and reliable answers.

This POC implements a RAG pipeline designed to address the distinct needs of two personas: engineers and marketers. Using LangChain within a Google Colab notebook, the system ingested arXiv papers, Wikipedia pages, and web content, creating a robust document store. The input question was vectorized using embedding models and compared to precomputed chunk embeddings to identify relevant passages. Outputs were customized to suit the specific needs of each persona, emphasizing technical precision for engineers and concise clarity for marketers.

The top-3 models had comprehensive scores of ∼3.7 and pairwise scores of ∼3.2. The system demonstrated differences in metric performance between the two personas, particularly among the pairwise scores. For the top-3 models, the marketing persona scored higher than the research persona.

run_number	avg_chunk_score	context_relevance	faithfulness	answer_relevance	pairwise_score	comprehensive
runB	0.52	4.33	3.41	3.85	3.21	3.7
runE	0.8	4.59	3.31	3.62	3.23	3.69
runF	0.8	4.5	3.35	3.77	3.21	3.71

Figure 1. The top-3 models had similar comprehensive scores of ∼ 3.7

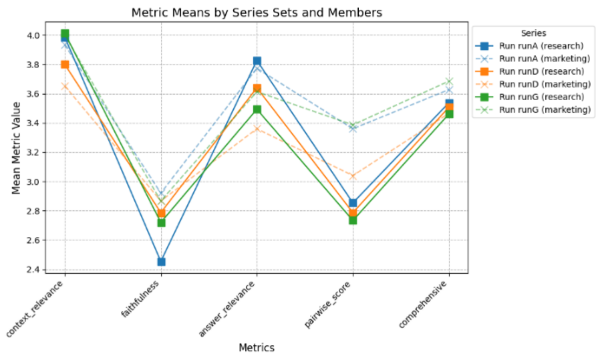

Figure 2. Variation across metrics for the top-3 models, by persona. The marketing persona had a higher pairwise score.

2 Key Findings

The best model specifications found incorporated both Cohere and Mistral as generative models, llama as a judge model, two embedding models, high chunk sizes, and a high number of retrieved chunks

- Higher chunk_size was associated with improved performance.
- Higher values of top_k lead to improved judge metrics.
- Both generative models were in high-scoring specifications.
- Persona-specific instructions were important for raising overall model performance. Pairwise score was strongly correlated with persona-specific instructions such as passage length.
- The prompt design had broad impact, and performance generally increased with more numerous and precise instructions.

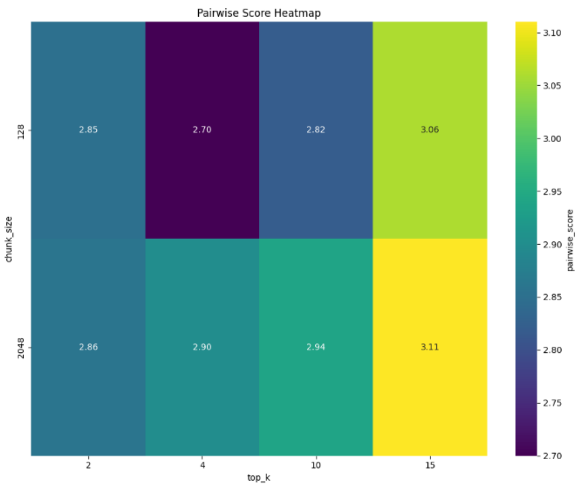

Figure 3. Higher top_k and chunk_size values were associated with higher comprehensive scores.

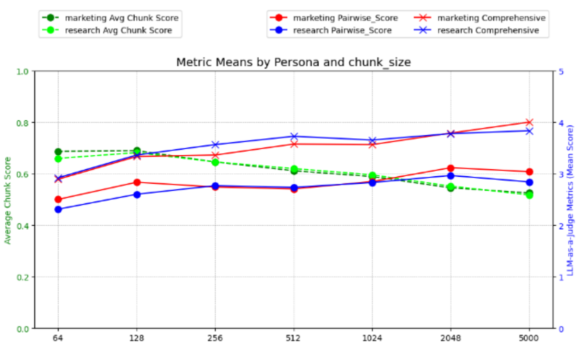

Figure 4. The best model saw better performance for the marketing persona. The pairwise score was consistently the lowest among the metrics scored by judge models.

In general, the marketing persona yielded higher metrics than the engineering persona.

After the final RAG prompt template was developed, it included over 20 lines, some of which were variable such as persona instructions. A systematic study was conducted to remove different components from the main prompt template.

Removal of persona-specific instructions (13B, 13K) for passage length reduced the pairwise score for both personas, but had an especially pronounced effect on the engineering persona.

Formatting embedded in the prompt, such as special begin/end tokens also had a big impact on the best Mistral model, decreasing the score by 0.2 from the baseline.

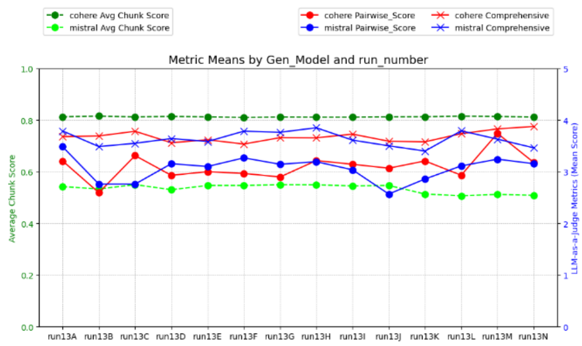

Figure 5. Changes in performance based on variation in prompt components.

3 Methods

3.1 Technical approach

The RAG pipeline used LangChain for chaining prompts and functions, integrating multiple LLMs and embedding models. The primary variables in the system were: persona, persona_instructions, question, gold_answer, generated_answer, context, and length. Tailored persona instructions differentiated technical density, language complexity, and answer length.

<u>Design choices:</u>

- Embedding models: mpnet and GIST embeddings were selected for their ability to capture semantic relations in technical and general-purpose text

- Chunking and overlap: Large chunks (2048 tokens) with 512-token overlaps ensured relevant context was included
- Prompt Engineering: a structured prompt template was devised to guide generative models for QA and judge models for scoring. Tailored instructions improved persona alignment.

Persona Accommodation:

- Engineers received longer, technically detailed answers with more acronyms and references, while marketing folks received shorter, high-graded responses. This customization ensured relevance and clarity for distinct audiences.

3.2 Testing and evaluation

Experiments were run with a subset of 15 randomly selected questions (comprising 20% of the overall set) for rapid data acquisition.

The mean retrieval score for the filtered chunks, computed as the default similarity score for the vectorstore, was measured and stored as avg_chunk_score. It is a measure of the vector similarity of the vectorized representations the input question with each chunk.

The RAGAS framework was used as inspiration for developing 3 metrics scored by a judge LLM.[3]

- Context Relevance - a measure of how closely the context aligns with the question.
- Faithfulness - a measure of how closely the answer aligns with the context.
- Answer Relevance - a measure of how closely the answer aligns with the question.

A prompt template was crafted for each metric, and considerable engineering of all 3 judge templates (along with suitable text parsing) was necessary to ensure the output was structured correctly. Since each of these three metrics is generated by an LLM, a Likert scale was specified as the generated score.

Since gold answers were available during model development, it was possible to measure the quality of the generated answers by making a pairwise comparison. Like the other judged metrics, a prompt template was engineered to provide the score and underwent several iterations.

Invalid metric scores were sometimes produced for the judged metrics but were minimized during model development, but a ~5% ratio of invalid scores could not be eliminated. Invalid scores were especially influenced by insufficient context length of judge model and were always excluded from statistical computations.

The mean score of the judged metrics was computed as a comprehensive representation of overall model performance, but unusual trends or outliers in other metrics were also noted. Prior to hyperparameter optimization and prompt engineering, the mean comprehensive score was less than 3 for the first experiment.

Medians and means were computed for all of the metrics. The best-performing models always had median pairwise scores of 4. Standard deviations were large given the Likert-nature of the judged scores, but means nevertheless were shown to vary reproducibly by the order of ~1.5 at increments of ~0.1 across the span of tested chunk_size. I therefore consider these mean Likerts to be reasonable metrics and differences of ~0.1 to be significant.

In addition, generated answers were visually inspected and compared with the gold answers to verify if the questions were correctly answered and that the content and style were matching. Generated answers that seemed excellent upon inspection were not always scored with high pairwise scores, demonstrating variability even for the best judge model.


```
    "25": {
        "question": "What benchmark did Chinchilla achieve an average accuracy of 67.5% on?",
        "gold_answer_research": "Chinchilla achieved an average accuracy of 67.5% on the MMLU benchmark
(Measuring Massive Multitask Language Understanding).",
        "gold_answer_marketing": "Chinchilla achieved an average accuracy of 67.5% on the MMLU benchmark
(Measuring Massive Multitask Language Understanding).",
        "research": {
            "generated_answer": "The Measuring Massive Multitask Language Understanding (MMLU) benchmark
is where Chinchilla achieved an average accuracy of 67.5%.",
            "avg_chunk_score": 0.7663070983451452,
            "pairwise_score": 4,
            "context_relevance_score": 4,
            "failthfulness_score": 2,
            "answer_relevance_score": 4,
            "comprehensive_score": 3.5
        },
        "marketing": {
            "generated_answer": "The Measuring Massive Multitask Language Understanding (MMLU) benchmark
is where Chinchilla achieved an average accuracy of 67.5%.",
            "avg_chunk_score": 0.7663070983451452,
            "pairwise_score": 5,
            "context_relevance_score": 3,
            "failthfulness_score": 5,
            "answer_relevance_score": 5,
            "comprehensive_score": 4.5
```


Figure 6. Example output showing a high-quality generated engineering response that nevertheless only scored a 4 for pairwise score.

4 Results and findings

4.1 Proof of concept functionality

The system successfully demonstrated RAG's potential to deliver persona-specific answers. Best-performing configurations achieved comprehensive scores near 3.7, exceeding initial baselines by over 20%. Enhanced chunking strategies and tailored prompts were critical to these improvements.

4.2 Lessons Learned

- Prompt Engineering Matters: Small prompt modifications had outsized impacts on performance metrics, especially for faithfulness and pairwise scores.
- Persona-Specific Optimization: Customizing outputs for different user types ensured relevance and usability.
- Embedding Model Selection: Embedding choice significantly affected chunk retrieval quality, with GIST outperforming MPNet in some configurations.
- Metric Reliability: Likert-scale metrics proved robust for comparative evaluation, though invalid scores highlighted the need for robust preprocessing.
- Scalability Considerations: Testing at scale required careful resource management, particularly with API limits and runtime constraints.

4.3 Challenges and limitations

~5 % of scores remained invalid due to judge model limitations such as insufficient context length (especially for Gemma) or inconsistent score formatting.

It is very clear that the prompt templates used for RAG as well as evaluation have a huge impact, and it would have been beneficial to know the specific requirements from the engineering and marketing teams prior to developing the templates. This was especially true since my original assumptions about the marketing output (i.e. that it should have a catchier style) yielded poor results.

For this POC, I required more calls to Cohere than were available with the trial key, so ultimately it was necessary to purchase a production key.

It is also clear that it's critical to carefully review the generated text. In my case, an early error in the context retrieval necessitated re-gathering baseline data.

4.4 Next steps

With more time, several other experiments could have been tried. It would have been interesting to devise a model that could dynamically shift between the generative model itself based on the specified persona. This approach could lead to improved overall performance since the best language model for each persona could be used.

Another experiment I was conducting, but did not complete, was a statistical evaluation of the metrics, given constant hyperparameters. One challenge with this study arose due to the Likert-data making mean scores less interpretable as well as difficulties in testing large enough samples for good Likert histograms.

Additional future work might involve exploring other generative models and perhaps even pretraining one on the provided gold dataset.

Summary and recommendations

This POC highlights the potential of RAG systems to deliver persona-specific, high quality answers tailored to distinct user needs. By iterating across 7 hyperparameters and engineering improved prompts (including 11 components of the main RAG prompt), the system achieved significant improvements in comprehensive scores, rising from an initial baseline of ~3.0 to ~3.7. This improvement underscores the importance of enriched contexts and detailed prompts for optimizing system performance.

Key takeaways:

- Persona-specific prompts and highly detailed instructions were critical, especially for the engineering persona, which showed higher faithfulness and pairwise scored when tailored prompts were employed.
- Larger enriched contexts (tunable via higher chunk sizes and number of concatenated chunks) yielded improved metrics and better answer quality.

- The combination of generative models (Cohere and Mistral) demonstrated complementary strengths, suggesting potential for a combined approach

Recommendations for RAG deployment:

- Deploy the system w/ a focus on tasks where persona alignment is critical, such as customer support
- Gather requirements from engineering and marketing teams to understand their particular needs and help to refine prompt templates

References

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Riedel, S. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *arXiv preprint arXiv:2005.11401.* Retrieved from https://arxiv.org/abs/2005.11401

Zheng, L., Chen, Z., Zou, Y., Lin, J., Liu, C., Yu, F., & Stoica, I. (2023). Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. *arXiv preprint arXiv:2306.05685.* Retrieved from https://arxiv.org/abs/2306.05685

Es, S., James, J., Espinosa-Anke, L., Schockaert, S. (2023). RAGAS: Automated Evaluation of Retrieval Augmented Generation. *arXiv preprint arXiv: 2309.15217.* Retrieved from https://arxiv.org/abs/ 2309.15217

ChatGPT was used when to critique the prompt templates for judge models during extensive iteration. It was also used to improve key sections of this report, especially Section 4.

A Appendices

```
# Run 13B - from Complete Prompt, remove persona-specific instructions for passage length
# Run 13C - "from Complete Prompt, remove persona-specifiv instructions for writing style
# Run 13D - from Complete Prompt, remove persona-specific instructions but keep persona reinforcement
# Run 13E - from Complete Prompt, remove line 4 & 10 -- remove persona-specific instructions and persona reinforcement
# Run 13F - from Complete Prompt, remove line 5 -- ignore specific content in the context
# Run 13G - from Complete Prompt, remove line 6 -- require answers to be retrieved from the context
# Run 13H - from Complete Prompt, remove line 8 -- reinforce best practice for insufficient context
# Run 13I - from Complete Prompt, remove line 9 -- require specific output formatting
# Run 13J - from Complete Prompt, remove line 11 -- reinforce a limited answer
# Run 13K - from Complete Prompt, swap lines 13/14 and 16/17 --  re-order {context} and {question}
# Run 13L - from Complete Prompt, remove line 20 -- "Assistant: "
# Run 13M - from Complete Prompt, remove [INST] and [/INST]
# Run 13N - from Complete Prompt, remove "Assistant: ", [INST] and [/INST]
# Run 13O - remove all detailed instructions, except keep persona reinforcement
# Run 13P - remove all detailed instructions
```

Table showing the list of components of the main RAG prompt template and the corresponding experiment number

```
rag_template = """[INST]
You work in the {persona} team at a startup company delivering Generative AI based products.
Review the provided context, then tailor your answer to the {persona} team.
You should ignore pieces of the context that are irrelevant formatting, such as html or unrelated
academic citations.
Use only the provided context to answer the question.
Identify the key information in the context that directly answers the question.
If the context alone does not provide enough information to answer the specific question, clearly
state that the topic isn't explicitly available in the context, then use the contextual information
to provide the closest possible answer.
Ensure your answer is provided as human-readable sentences.  It should not include excessive
formatting or lists.
{persona_instructions}
Your answer should address the provided question without including extraneous or inpertinent
information.

### Context:
{context}

### Question:
{question}

\n[/INST]
Assistant: """
rag_prompt = PromptTemplate(
        template=rag_template,
        input_variables=["persona", "persona_instructions", "context", "question"]
)




# For all metrics Prompts, consider updating this prompt to include Chain-of-Thought-Reasoning, then
a final keyword (eg. "FINAL SCORE: ") so that the final number can be parsed
```

Prompt template for RAG process.

```
pairwise_score_template = """[INST]
You are an amazing judge capable of issuing a final numerical score.
Compare the following two answers to the provided question.  Carefully evaluate the Generated Answer
against the Gold Answer based on these criteria:
1. Semantic similarity:  Does the Generated Answer convey the same meaning as the Gold Answer?
2. Completeness:  Does the Generated Answer include all of the relevant information from the Gold
Answer?
3. Alignment with the Question:  Does the Generated Answer address the question appropriately?
4. Passage length:  Does the Generated Answer have the same length as the Gold Answer?

Provide a brief evaluation of these criteria, followed by an integer score on a scale of 1 to 5:
- 5: Near-identical answers
- 4: Minor differences that do not impact meaning
- 3: Moderate differences that partially impact meaning
- 2: Major differences that significantly impact meaning
- 1: Completely different answers

Here is an Example:

* Example Question:
What is the significance of the band gap in semiconductors?

* Example Gold Answer:
The band gap in semiconductors is the energy difference between the valence band and the conduction
band. It determines the electrical and optical properties of the material, including its ability to
conduct electricity under certain conditions.

* Example Generated Answer with high comparison score:
The band gap is the energy difference between the valence and conduction bands in semiconductors,
affecting their electrical and optical properties.

* Example Generated Answer with low comparison score:
A semiconductor's band gap, which is closely related to the HOMO/LUMO gap in chemistry, varies as a
function of temperature as described by Varshni.

Now please provide a context relevance score for the following context, given the following question:

* Question:
{question}

* Gold Answer:
{gold_answer}

* Generated Answer:
{generated_answer}

First, think step by step to evaluate the criteria and summarize your reasoning.
Conclude by stating your final score with "FINAL ANSWER: X" (where X is an integer score between 1
and 5).

\n[/INST]
Assistant: """
pairwise_score_prompt = PromptTemplate(
        template=pairwise_score_template,
        input_variables=["question", "gold_answer", "generated_answer"]
)
```

Prompt template for pairwise score

```
context_relevance_score_template = """[INST]
You are an amazing judge capable of issuing a final numerical score.
Evaluate the relevance of the following context to the given question based on these criteria:
1. Topical alignment:  Does the context address the main topic of the question?
2. Specificity:  Does the context include information directly answering the question?
3. Lack of irrelevant material:  Does the context avoid unrelated or distracting content?

Provide a brief evaluation of these criteria, followed by an integer score on a scale of 1 to 5:
- 5: The context is fully relevant to the question, with no irrelevant material
- 4: Mostly relevant, but includes some minor irrelevant material
- 3: Moderately relevant, with significant irrelevant content
- 2: Minimally relevant, with primarily irrelevant content
- 1: Entirely irrelevant

Here is an Example:

* Example Question:
When was the Chimnabai Clock Tower completed, and who was it named after?

* Example Context with high context relevance:
The Chimnabai Clock Tower, also known as the Raopura Tower, is a clock tower situated in the Raopura
area of Vadodara, Gujarat, India. It was completed in 1896 and named in memory of Chimnabai I (1864-
1885), a queen and the first wife of Sayajirao Gaekwad III of Baroda State.

* Example Context with low context relevance:
The Chimnabai Clock Tower, also known as the Raopura Tower, is a clock tower situated in the Raopura
area of Vadodara, Gujarat, India. It was completed in 1896 and named in memory of Chimnabai I (1864-
1885), a queen and the first wife of Sayajirao Gaekwad III of Baroda State. It was built in Indo-
Saracenic architecture style. History. Chimnabai Clock Tower was built in 1896. The tower was named
after Chimnabai I (1864-1885), a queen and the first wife of Sayajirao Gaekwad III of Baroda State.
It was inaugurated by Mir Kamaluddin Hussainkhan, the last Nawab of Baroda. During the rule of
Gaekwad, it was a stoppage for horse drawn trams. The clock tower was erected at the cost of 25,000
(equivalent to 9.2 million or USD 120,000 in 2023).

Now please provide a context relevance score for the following context, given the following question:

* Question:
{question}

* Context:
{context}

Think step by step to evaluate the criteria and summarize your reasoning.
Conclude by stating your final score with "FINAL ANSWER: X" (where X is an integer score between 1
and 5).

\n[/INST]
Assistant: """
context_relevance_score_prompt = PromptTemplate(
        template=context_relevance_score_template,
        input_variables=["context", "question"]
)
```

Prompt template for context relevance score

```
faithfulness_score_template = """[INST]
You are an amazing judge capable of issuing a final numerical score.
Evaluate the faithfulness of the following context to the given answer based on these criteria:
1. Factual accuracy: Is the information in the answer factually consistent with the context?
2. Alignment: Does the answer use information directly from the context?
3. Completeness: Does the context support all key claims made in the answer?

Provide a brief evaluation of these criteria, followed by an integer score on a scale of 1 to 5:
- 5: Fully faithful, where all claims in the answer are consistent with the context
- 4: Mostly faithful, with minor inconsistencies
- 3: Moderately faithful, where some key claims lack support or are inconsistent
- 2: Minimally faithful, where most claims lack support or are inconsistent
- 1: Completely unfaithful, where the answer is entirely unsupported by the context

Here is an Example:

* Example Context:
Oppenheimer is a 2023 biographical thriller film written and directed by Christopher Nolan. Based on
the 2005 biography American Prometheus by Kai Bird and Mar tin J. Sherwin, the film chronicles the
life of J. Robert Oppenheimer, a theoretical physicist who was pivotal in developing the first
nuclear weapons as part of the Man hattan Project, and thereby ushering in the Atomic Age.  Cillian
Murphy stars as Oppenheimer, with Emily Blunt as Oppenheimer's wife Katherine "Kitty" Oppenheimer.

* Example Answer with high faithfulness:
Christopher Nolan directed the film Oppenheimer. Cillian Murphy stars as J. Robert Oppenheimer in the
film.

* Example Answer with low faithfulness:
James Cameron directed the film Oppenheimer. Tom Cruise stars as J. Robert Oppenheimer in the film

Now please provide a faithfulness score for the following answer, given the following context.

* Context:
{context}

* Answer:
{generated_answer}

Think step by step to evaluate the criteria and summarize your reasoning.
Conclude by stating your final score with "FINAL ANSWER: X" (where X is an integer score between 1
and 5).

\n[/INST]
Assistant: """
faithfulness_score_prompt = PromptTemplate(
        template=faithfulness_score_template,
        input_variables=["context", "generated_answer"]
)
```

Prompt template for faithfulness score

```
answer_relevance_score_template = """[INST]
You are an amazing judge capable of issuing a final numerical score.
Evaluate the relevance of the following answer to the given question based on these criteria:
1. Alignment:  Does the answer directly address the question?
2. Completeness:  Does the answer include sufficient detail to fully answer the question?
3. Relevance:  Does the answer avoid including irrelevant or tangential information?

Provide a brief evaluation of these criteria, followed by an integer score on a scale of 1 to 5:
- 5: Fully relevant, with an answer that directly and completely answers the question
- 4: Mostly relevant, with minor omissions or tangential content
- 3: Moderately relevant, which partially answers the question
- 2: Minimally relevant, which largely fails to answer the question
- 1: Completely irrelevant, since the answer does not address the question at all

Here is an Example:

* Example Question:
When is the scheduled launch date and time for the PSLV-C56 mission, and where will it be launched
from?

* Example Answer with high answer relevance:
The PSLV-C56 mission is scheduled to be launched on Sunday, 30 July 2023 at 06:30 IST / 01:00 UTC. It
will be launched from the Satish Dhawan Space Centre, Sriharikota, Andhra Pradesh, India.

* Example Answer with low answer relevance:
The scheduled launch date and time for the PSLV-C56 mission have not been provided.The PSLV-C56
mission is an important space mission for India. It aims to launch a satellite into orbit to study
weather patterns.

Now please provide a faithfulness score for the following answer, given the following question:

* Question:
{question}

* Answer:
{generated_answer}

Think step by step to evaluate the criteria and summarize your reasoning.
Conclude by stating your final score with "FINAL ANSWER: X" (where X is an integer score between 1
and 5).

\n[/INST]
Assistant: """
answer_relevance_score_prompt = PromptTemplate(
        template=answer_relevance_score_template,
        input_variables=["generated_answer", "question"]
)
```

Prompt template for answer relevance score