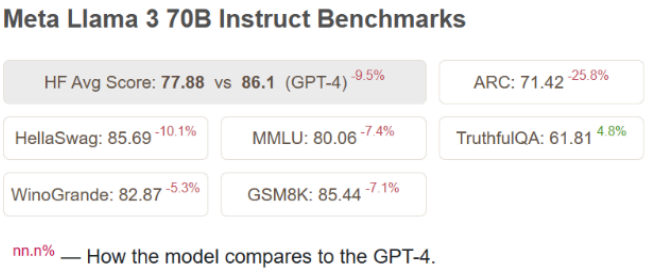
LLM-Quiz-Wizard - Summary Document:

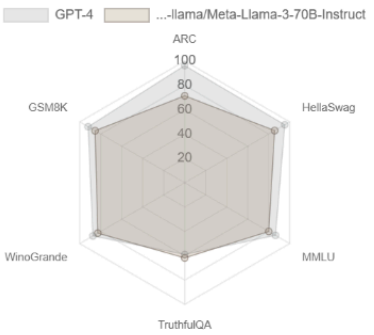
The LLM-Quiz-Wizard is an LLM-powered Streamlit-based web application designed to generate quiz questions dynamically. The app is deployed online and fully documented in a GitHub repository with installation and running instructions.

Selected Tools:

1. **Groq:** A computing engine optimized for fast AI workloads. Its LPU engine ensures fast and efficient processing of large language models, significantly reducing latency and enhancing the responsiveness of the quiz generation application.
2. **Llama-3**: A relatively new cutting-edge, state-of-the-art large language model known for its advanced natural language understanding and generation capabilities. I chose this LLM for its accuracy and ability to generate coherent, contextually relevant questions. Its large model size ensures a deep semantic representation of varied topics, making quizzes engaging and challenging.

   The model reaches high performance, close to GPT-4 on several benchmarks, as shown in the following figures from LLM Explorer.





3. **Streamlit**: The application's UI is built using Streamlit for a user-friendly interface.

Implementation Details:

- **Main Application (app.py)**: The main UI script.
- **LLM Interaction (llm_operator.py)**: Manages interactions with groq Llama-3, ensuring smooth question generation.
- **Prompt Management (prompts.py)**: Contains functions for creating and customizing prompts sent to the language models.

Prompt engineering techniques I used:

1. **Few-shot learning:** Using detailed examples of prompts and answers within the prompt. By providing explicit examples of user inputs and corresponding responses, the LLM uses the additional context to understand the expected response
2. **Role Definition**: The system prompt defines the AI as a "trivia writing expert" with access to various trivia sources, ensuring it leverages relevant domain associations to craft questions.
3. **Clear Input Requirements**: The user can specify "Topic," "Number of questions," and "Difficulty" level. The `make_user_prompt()` function tailors instructions based on the difficulty level. It provides specific guidelines for each difficulty level.
4. **Validation and Quality Assurance**: The prompt emphasizes the importance of correctness, ensuring “fact-checked", diverse, and engaging questions.
5. **Structured Output**: A strict JSON format is specified for the output, detailing fields like "general_topic," "sub_topic," "difficulty," "question," "options," "answer," and "answer explanation." This structure ensures consistency and ease of use.

These techniques generate well-rounded, accurate, engaging quiz questions tailored to the

Potential enhancements:

- Storing user data in a database for performance analytics, customizable themes, gamification, collaborative quiz sharing, etc.
- Contextual Layering - Incorporate additional context, such as recent news or user preferences, to make questions more relevant and dynamic (e.g., RAG).
- Develop prompts for follow-up questions, creating a more interactive and conversational quiz experience.
- Use real-time user feedback to refine and improve prompt quality continuously
- Additional question types and hint system