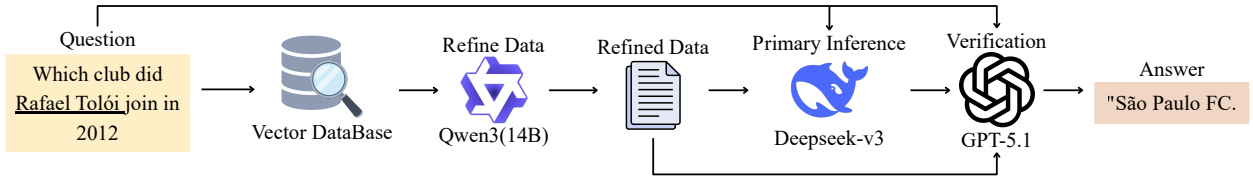
SetPieceRAG: Domain-Specific RAG for Knowledge-Intensive Soccer VQA with Large Language Models

Supplementary Material

7. Task-Specific Pipeline Details

Q1. Background Knowledge Text QA



Overview of the proposed pipeline.

Refine Data (Qwen3 Prompt Template)

```
You are an elite soccer data extraction engine. Your ONLY job is to extract facts from the [Context] that directly help evaluate the [Options] for the given [Question].

[Context from SoccerWiki]:
{wiki_text}

[Question]:
{q_text}

[Options]:
{options_str}

[Strict Instructions]:
1. Extract ONLY the exact facts (dates, teams, appearances, goals, awards) relevant to the Question and Options.
2. Format the output as a clean, concise bulleted list.
3. Convert raw table data (e.g., '2016-2018: EspanyolB 37 (13)') into clear natural language sentences (e.g., '- Made 37 appearances for Espanyol B between 2016 and 2018.').
4. DO NOT answer the question directly. DO NOT add any outside knowledge. DO NOT include filler (Never output phrases like "Organized Facts:" or "Here is the summary").
5. If the [Context] contains NO relevant information, output exactly: "NO_RELEVANT_INFO".

Extraction:
```

Primary Inference (DeepSeek Prompt Template)

You are an expert soccer data evaluator.
I have attached a JSON file containing question IDs, questions, multiple-choice options, and precise facts extracted from a trusted database
under the key `slm_organized_context`.

[Instructions]
1. You MUST use the `slm_organized_context` from the attached file as the absolute ground truth to deduce the correct option.
2. If the `slm_organized_context` is exactly "NO_RELEVANT_INFO", you must rely entirely on your own internal soccer knowledge to find the correct answer.
3. Output the final result strictly in the format shown below. DO NOT output any explanations, reasoning, conversational filler, or extra text.

[Output Format Example]
- ID 2: O2
- ID 5: O1
- ID 6: O4

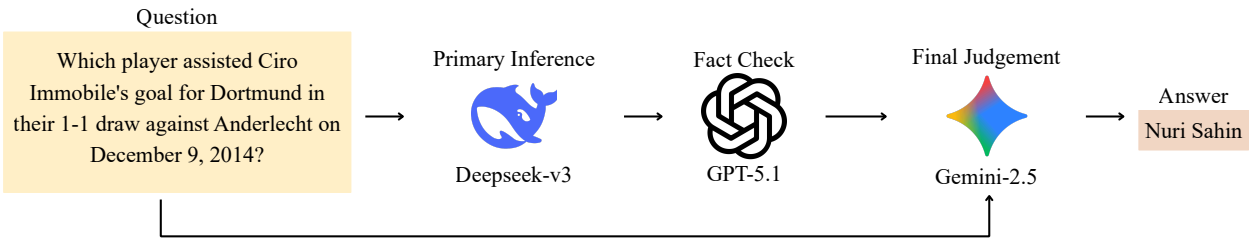
Verification (GPT Prompt Template)

[Task] Verify the soccer data and determine the final correct option (O1, O2, O3, or O4).

[Question]: {question}
[Options]: {options}
[SLM Context (Source 1)]: {context}
[DeepSeek's Raw Reasoning (Source 2)]: {raw_reasoning}

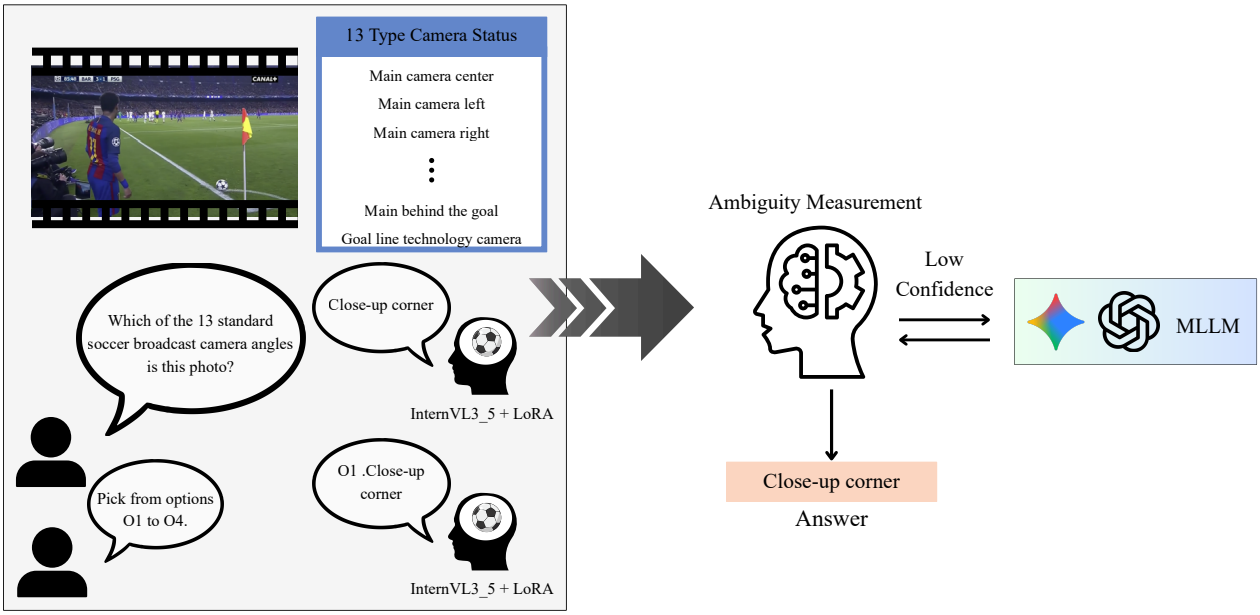
[Instructions]
1. Read DeepSeek's Raw Reasoning. It might contain dates in different formats (e.g., "10 21" for "October 21") or semantic answers (e.g., "football"). Match them logically with the provided [Options].
2. Use the [SLM Context] as primary evidence.
3. If the context is "NO_RELEVANT_INFO" or you find any ambiguity, you MUST use web search to find the exact historical fact.
4. Correct any errors found in DeepSeek's logic.
5. Output ONLY a clean JSON object with the following keys:
   - "final_answer": (The option key like "O1", "O2", etc.)
   - "reasoning": (Short explanation of your verification)

Q2. Match Situation QA



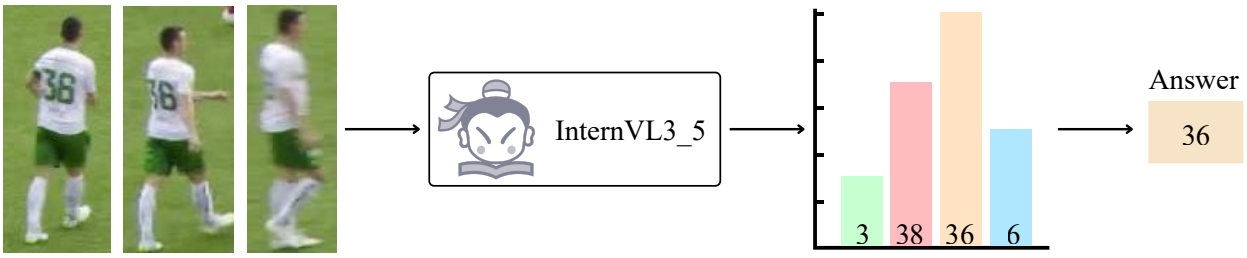
Overview of the proposed pipeline.

Q3. Camera Status Classification



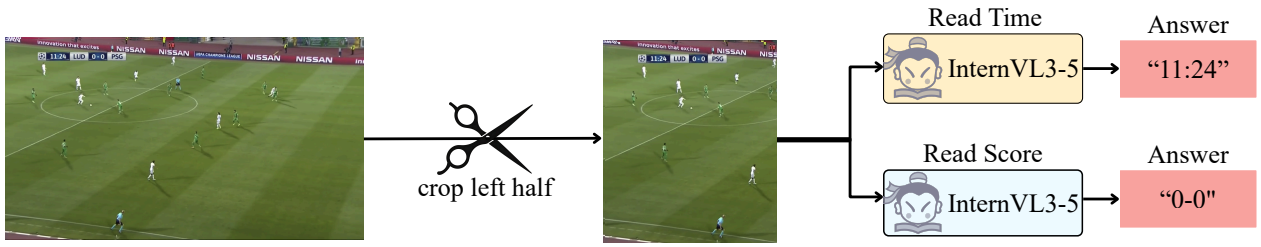
Overview of the proposed pipeline.

Q5. Jersey Number Recognition



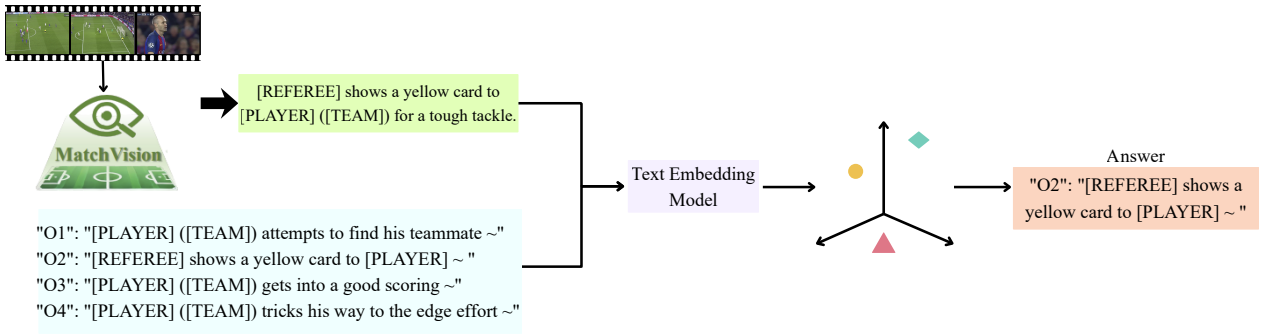
Overview of the proposed pipeline.

Q6. Score and Time Relevant QA



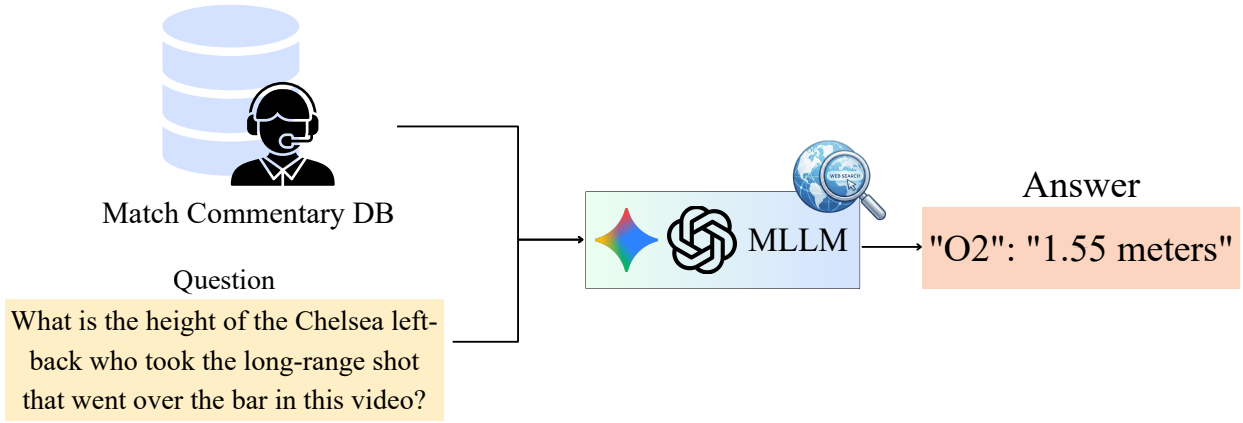
Overview of the proposed pipeline.

Q11. Commentary Generation



Overview of the proposed pipeline.

Q12. Commentary Relevant QA



Overview of the proposed pipeline.

8. Implementation Details for SetPieceRAG

8.1. Visual Index Construction.

All player images are sourced from the SoccerWiki [32] dataset hosted on Hugging Face. Each image is converted to RGB and encoded through a frozen CLIP ViT-B/32 model to produce a 512-dimensional embedding. Each embedding is L2-normalized and added to a FAISS IndexFlatIP index. The final index contains 27,928 entries. Each entry is associated with a cleaned player name derived from the source folder name by removing the trailing unique identifier (e.g., `Aaron_Connolly_4f5o2me0` → `Aaron Connolly`). The same name-cleaning logic is applied consistently across both the visual index and the text knowledge base to ensure key alignment.

8.2. Text Knowledge Base Construction.

Player textual knowledge is sourced from individual JSON files in the SoccerWiki [32] dataset. Each JSON file contains structured profile attributes such as career history, team membership, and match statistics. No text chunking or segmentation is applied; the entire JSON record is preserved as a single entry and stored in a dictionary keyed by the cleaned player name.

8.3. Retrieval and Inference.

At inference, the input image is encoded by the same frozen CLIP ViT-B/32 and L2-normalized. The top-1 nearest neighbor (k=1) is retrieved from the FAISS index via inner product similarity. The retrieved player name is used to look up the corresponding JSON record from the text knowledge base. If no matching record is found, the context is marked as unavailable. The retrieved context, question, and four multiple-choice options are concatenated into a structured prompt with an explicit JSON-format constraint requiring both a reasoning field and an answer field.

8.4. Tier-2 Fallback Conditions.

The Gemini [36] fallback is triggered when any of the following conditions are met: (i) the Tier-1 model output contains error indicators or uncertainty expressions, (ii) the retrieved knowledge base context is unavailable for the identified player, or (iii) the generated output does not contain a valid answer token (O1–O4). Upon activation, Gemini [36] is invoked with the Google Search tool enabled and the same JSON-format constraint applied.

8.5. Answer Extraction.

The system applies a multi-stage answer extraction procedure to ensure robust parsing across different LLM backends: first, the output is parsed as JSON to extract the answer field; if parsing fails, a regex pattern matching O1–O4 is applied.

8.6. Computational Cost.

All experiments were conducted on a single NVIDIA A100 (80GB) in Google Colab Pro+. For Q4 (Background Knowledge Image QA, 50 questions), Tier-1 (LLaMA 3.2 1B) inference completes in under 1 second per query. When Tier-2 (Gemini API with Google Search) is triggered, which occurs in 88% of Q4 queries, latency increases to approximately 5–10 seconds per query due to live web search. Web search results are not cached between evaluation runs, and evaluation was conducted at a fixed point in time (February 2026) to minimize variability from real-time web content. For tasks that do not require Tier-2 fallback, the system operates entirely locally with minimal API cost.