

Fantasy Premier League Trivia Assistant

Advanced Computer Lab

Presented by:

Habiba Elsonbaty

Zeina Fadel

Abdelrahman Elaby

Ehab Medhat



Project Overview & Agenda



High-Level System Architecture

Understanding the core pipeline for our FPL AI Assistant.



Input Preprocessing

Intent classification and entity extraction.



Graph Retrieval Layer: Baseline

Cypher query templates and initial data retrieval.



Graph Retrieval Layer: Embedding-Based

Leveraging advanced embeddings for richer context.



LLM Layer & Comparison

Context construction, prompt engineering, and LLM evaluations.



Remaining Limitations

Future recommendations and improvements

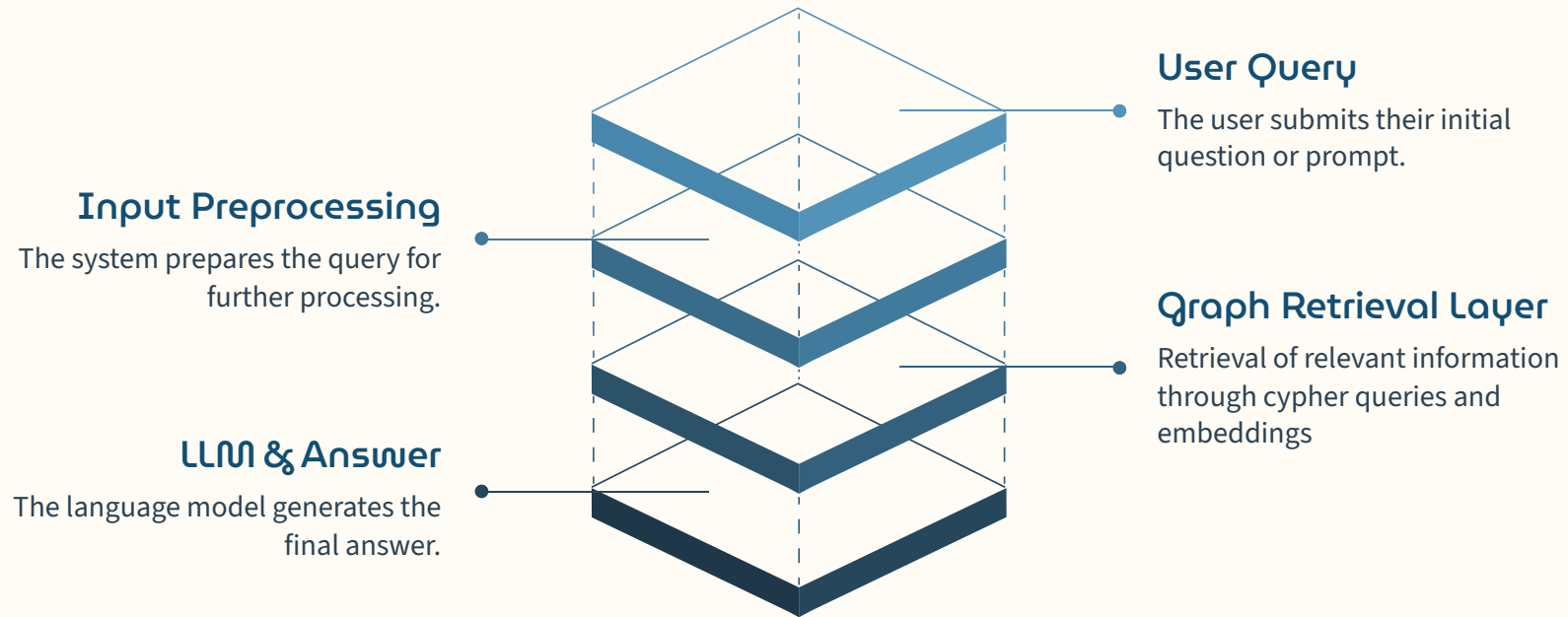


Live Demonstration

See the FPL AI Assistant in action.

High-Level System Architecture

Our Fantasy Premier League (FPL) AI Assistant processes natural language queries to provide insightful, data-driven answers. The core pipeline integrates intent classification, entity extraction, graph database retrieval, and Large Language Models (LLMs) to deliver a comprehensive FPL analysis.



Dataset Added Changes



Added Team to represent each player's current team



Generated relation `PLAYS_FOR` in KQ linking between a player and his team



Calculated cumulative sums of most important stats up to each fixture to aid the LLM in totaling and summing queries

team	total_points_sum	goals_scored_sum	assists_sum	minutes_sum	bonus_sum	clean_sheets_sum
Brighton	0	0	0	0	0	0
Brighton	1	0	0	45	0	0
Brighton	1	0	0	45	0	0
Brighton	1	0	0	45	0	0
Brighton	1	0	0	45	0	0
Brighton	2	0	0	59	0	0
Brighton	2	0	0	59	0	0
Brighton	2	0	0	59	0	0
Brighton	2	0	0	59	0	0
Brighton	2	0	0	59	0	0
Brighton	2	0	0	59	0	0
Brighton	2	0	0	59	0	0
Brighton	2	0	0	59	0	0
Brighton	2	0	0	59	0	0
Brighton	2	0	0	59	0	0
Brighton	3	0	0	94	0	0
Brighton	5	0	0	154	0	0
Brighton	5	0	0	154	0	0
Brighton	5	0	0	154	0	0
Brighton	5	0	0	154	0	0
Brighton	5	0	0	154	0	0

Intent Classes

Fixture Details

Retrieve upcoming matches, kickoff times, and fixture schedules.

Best Players by Metric

Identify top-performing players ranked by statistical metrics.

Worst Players by Metric

Identify lowest-performing players ranked by statistical metrics.

Player or Team Performance

Compare and analyze performance statistics between players or teams.

Player Information

Retrieve profile details about players (Position and Team).

Rule Based Approach

1

Look for **keywords and patterns** associated with each intent.

2

Assign **weighted scores** to each intent based on matched patterns.

3

Select the intent **with the highest score** as the final classification.

4

Default to **player/team performance** if no clear intent is detected.

Example:

Prompt: When does Harry Kane play against Liverpool?

Rule-based Intent: fixture_details

Why not LLM?

1

Tried the facebook/bart-large-mnli

2

Misclassified queries with rare or specific FPL terminology.

3

Sensitive to minor changes in phrasing.

```
test = ["Who are the top 3 midfielders by assists?"]
for prompt in test:
    intent1 = classify_fpl_intents(prompt)
    intent2 = classify_intent_classifier(prompt)
    print(f"Prompt: {prompt}")
    print(f"Rule-based Intent: {intent1}")
    print(f"Classifier Intent: {intent2}")
    print()
```

✓ 50.7s

Prompt: Who are the top 3 midfielders by assists?
Rule-based Intent: best_players_by_metric
Classifier Intent: player_information

Entity Extraction Approach

1

Knowledge Base Lookup

- Load players, teams, positions, and stats from FPL database.
- Maintain a lookup dictionary for quick matching.

2

Text Preprocessing

- Normalize query text using spaCy.
- Tokenize and lemmatize for flexible matching.

3

Entity Matching

- Match query tokens against lookup dictionary.
- Handle multi-word entities (e.g., "Mohamed Salah", "Manchester City").

1

Regex & Pattern Extraction

- Extract structured info like gameweeks, seasons, top/bottom N players, and filters.

2

Defaults & Fallbacks

- Default stat type: `total_points`.
- Default season: `2022-23`.
- Default limit: 10 results.

Example:

Prompt: When does Harry Kane play against Liverpool?

Rule-Based Extracted Entities:

```
{'stat_type': 'total_points', 'season': '2022-23', 'limit': 10, 'team1': 'Liverpool', 'team_name': 'Liverpool', 'player1': 'Harry Kane', 'player_name': 'Harry Kane'}
```


Why This Works Better Than Standard NER

Struggles with unknown or rare player names and teams.

Often misses domain-specific stats and positions.

Unlike Rule Based Approach, it doesn't use FPL knowledge base for accurate matching.

```
test_entity=["Show me how many goals aboubacar traoré scored in gameweek 5"]
for text in test_entity:
    print("Prompt:", text)

    ner_entities = extract_entities_ner(text)
    rule_entities = extract_fpl_entities(text)

    print("\nNER Extracted Entities:")
    print(ner_entities.get("players"))

    print("\nRule-Based Extracted Entities:")
    print(rule_entities.get("player1"))
```

✓ 0.2s

Prompt: Show me how many goals aboubacar traoré scored in gameweek 5

NER Extracted Entities:

[]

Rule-Based Extracted Entities:

Boubacar Traoré

Graph Retrieval Layer: Baseline Cypher



Identify intent

(performance, fixtures, best/worst players, player info).



Use extracted entities

(player, team, stat, season, gameweek, limit).



Fill predefined Cypher query templates

with entity values.

Prompt: When does Harry Kane play against Liverpool?

Cypher Query:

```
MATCH (:Season {season_name:'2022-23'})-[:HAS_GW]->(g)-[:HAS_FIXTURE]->(f:Fixture)
MATCH (p:Player {player_name:'Harry Kane'})-[pi:PLAYED_IN]->(f)
MATCH (f)-[:HAS_HOME_TEAM]->(h:Team),
      (f)-[:HAS_AWAY_TEAM]->(a:Team)
MATCH (p)-[:PLAYS_FOR]->(t:Team)
WHERE h.name = 'Liverpool' OR a.name = 'Liverpool'
RETURN p.player_name AS player, t.name AS team, f.fixture_number AS fixture, f.kickoff_time, g.GW_number AS gameweek, h, a
ORDER BY f.kickoff_time ASC
```

Query Example

```
if intent == "player_or_team_performance" and player1 and not player2:
    if gw:
        return f"""
            MATCH (:Season {{season_name:'{season}'}})-[:HAS_GW]->(g:Gameweek {{GW_number:{gw}}})-[:HAS_FIXTURE]->(f:Fixture)
            MATCH (p:Player {{player_name:'{player1}'}})-[pi:PLAYED_IN]->(f)
            RETURN p.player_name AS player, pi.{stat} AS {stat}, g.GW_number AS gameweek

            """
    else:
        return f"""
            MATCH (:Season {{season_name:'{season}'}})-[:HAS_GW]->(g)-[:HAS_FIXTURE]->(f)
            MATCH (p:Player {{player_name:'{player1}'}})-[pi:PLAYED_IN]->(f)
            RETURN p.player_name AS player, SUM(pi.{stat}) AS total_{stat}, '{season}' AS season

            """
```

```

if intent == "player_or_team_performance" and team1 and not team2:
    if gw:
        # Single Gameweek performance for a team
        return f"""
            MATCH (:Season {{season_name:'{season}'}})-[:HAS_GW]->(g:Gameweek {{GW_number:{gw}}})-[:HAS_FIXTURE]->(f)
            MATCH (p:Player)-[:PLAYS_FOR]->(t:Team{{name:'{team1}'}})
            MATCH (p)-[pi:PLAYED_IN]->(f)
            RETURN t.name AS team, SUM(pi.{stat}) AS {stat}, g.GW_number AS gameweek
        """
    else:
        # Full season performance for a team
        return f"""
            MATCH (:Season {{season_name:'{season}'}})-[:HAS_GW]->(:Gameweek)-[:HAS_FIXTURE]->(f)
            MATCH (p:Player)-[:PLAYS_FOR]->(t:Team{{name:'{team1}'}})
            MATCH (p)-[pi:PLAYED_IN]->(f)

            RETURN t.name AS team, SUM(pi.{stat}) AS {stat}
        """

```

if intent == "player_or_team_performance" and player1 and player2:

if gw:

return f"""

UNWIND ['{player1}', '{player2}'] AS pname

MATCH (p:Player {{player_name: pname}})

OPTIONAL MATCH (p)-[pi:PLAYED_IN]->(f:Fixture)<-[:HAS_FIXTURE]-(g:Gameweek {{GW_number:{gw}}})<-[:HAS_GW]-(:Season {{season_name:'{season}'}})

RETURN

p.player_name AS player,

COALESCE(SUM(pi.{stat}), 0) AS total_{stat},

{gw} AS gameweek

ORDER BY player

"""

else:

return f"""

UNWIND ['{player1}', '{player2}'] AS pname

MATCH (p:Player {{player_name: pname}})

OPTIONAL MATCH (p)-[pi:PLAYED_IN]->(:Fixture)<-[:HAS_FIXTURE]-(g:Gameweek)<-[:HAS_GW]-(:Season {{season_name:'{season}'}})

RETURN

p.player_name AS player,

COALESCE(SUM(pi.{stat}), 0) AS total_{stat}

ORDER BY player

"""

```

if intent == "player_or_team_performance" and team1 and team2:
    if gw:
        return f"""
            MATCH (:Season {{season_name:'{season}'}})-[:HAS_GW]->(g:Gameweek {{GW_number:{gw}}})-[:HAS_FIXTURE]->(f
            MATCH (p:Player)-[pi:PLAYED_IN]->(f)
            MATCH (p)-[:PLAYS_FOR]->(t:Team)
            WHERE t.name = '{team1}' OR t.name = '{team2}'
            RETURN
            CASE
                WHEN t.name = '{team1}' THEN '{team1}'
                ELSE '{team2}'
            END AS team,
            SUM(pi.{stat}) AS total_{stat},
            g.GW_number AS gameweek
        """
    else:
        return f"""
            MATCH (:Season {{season_name:'{season}'}})-[:HAS_GW]->(:Gameweek)-[:HAS_FIXTURE]->(f)
            MATCH (p:Player)-[pi:PLAYED_IN]->(f)
            MATCH (p)-[:PLAYS_FOR]->(t:Team)
            WHERE t.name = '{team1}' OR t.name = '{team2}'
            RETURN
            CASE
                WHEN t.name = '{team1}' THEN '{team1}'
                ELSE '{team2}'
            END AS team,
            SUM(pi.{stat}) AS total_{stat}
        """

```

```

if intent == "fixture_details" and team1 and team2 and gw:
    return f"""
        MATCH (:Season {{season_name:'{season}'}}
        -[:HAS_GW]->(g:Gameweek {{GW_number:{gw}}})
        -[:HAS_FIXTURE]->(f:Fixture)
        MATCH (f)-[:HAS_HOME_TEAM]->(home:Team)
        MATCH (f)-[:HAS_AWAY_TEAM]->(away:Team)
        WHERE (home.name = '{team1}' AND away.name = '{team2}')
        OR (home.name = '{team2}' AND away.name = '{team1}')
        RETURN
            f.fixture_number AS fixture,
            f.kickoff_time AS kickoff_time,
            home.name AS home_team,
            away.name AS away_team
        ORDER BY f.kickoff_time ASC
    """

```

```

if intent == "fixture_details" and team1 and team2 and gw:
    return f"""
        MATCH (:Season {{season_name:'{season}'}}
        -[:HAS_GW]->(g:Gameweek {{GW_number:{gw}}})
        -[:HAS_FIXTURE]->(f:Fixture)
        MATCH (f)-[:HAS_HOME_TEAM]->(home:Team)
        MATCH (f)-[:HAS_AWAY_TEAM]->(away:Team)
        WHERE (home.name = '{team1}' AND away.name = '{team2}')
        OR (home.name = '{team2}' AND away.name = '{team1}')
        RETURN
            f.fixture_number AS fixture,
            f.kickoff_time AS kickoff_time,
            home.name AS home_team,
            away.name AS away_team
        ORDER BY f.kickoff_time ASC
    """

```

```
if intent == "fixture_details" and team1 and team2 and not gw:
```

```
    return f"""
```

```
        MATCH (:Season {{season_name:'{season}'}})
```

```
        -[:HAS_GW]->(g)
```

```
        -[:HAS_FIXTURE]->(f:Fixture)
```

```
        MATCH (f)-[:HAS_HOME_TEAM]->(home:Team)
```

```
        MATCH (f)-[:HAS_AWAY_TEAM]->(away:Team)
```

```
        WHERE (home.name = '{team1}' AND away.name = '{team2}')
```

```
        OR (home.name = '{team2}' AND away.name = '{team1}')
```

```
        RETURN
```

```
            f.fixture_number AS fixture,
```

```
            f.kickoff_time AS kickoff_time,
```

```
            g.GW_number AS gameweek,
```

```
            home.name AS home_team,
```

```
            away.name AS away_team
```

```
        ORDER BY f.kickoff_time ASC
```

```
    """
```

```
if intent == "best_players_by_metric" and not entities.get("position"):
```

```
    if entities.get("filter_value")!= None:
```

```
        val = entities["filter_value"]
```

```
        return f"""
```

```
            MATCH (:Season {{season_name:'{season}'}})-[:HAS_GW]->(g)-[:HAS_FIXTURE]->(f)
```

```
            MATCH (p:Player)-[:pi:PLAYED_IN]->(f)
```

```
            WITH p, SUM(pi.{stat}) AS total_stat
```

```
            WHERE total_stat > {val}
```

```
            RETURN p.player_name AS player, total_stat
```

```
            ORDER BY total_stat DESC LIMIT {limit}
```

```
        """
```


#continuation of past intent query formation

else:

```
return f"""
    MATCH (:Season {{season_name:'{season}'}})-[:HAS_GW]->(g)-[:HAS_FIXTURE]->(f)
    MATCH (p:Player)-[pi:PLAYED_IN]->(f)
    RETURN p.player_name AS player, SUM(pi.{stat}) AS total_{stat}
    ORDER BY total_{stat} DESC LIMIT {limit}
    """
```

if intent == "best_players_by_metric" and entities.get("position"):

position = entities["position"]

if entities.get("filter_value")!= None:

val = entities["filter_value"]

```
return f"""
    MATCH (p:Player)-[:PLAYS_AS]->(pos:Position {{name:'{position}'}})
    MATCH (:Season {{season_name:'{season}'}})-[:HAS_GW]->(g)-[:HAS_FIXTURE]->(f)
    MATCH (p)-[pi:PLAYED_IN]->(f)
    WITH p, pos, SUM(pi.{stat}) AS total_stat
    WHERE total_stat > {val}
    RETURN p.player_name AS player, total_stat, pos.name AS position
    ORDER BY total_stat DESC LIMIT {limit}
    """
```

else:

```
return f"""
    MATCH (p:Player)-[:PLAYS_AS]->(pos:Position {{name:'{position}'}})
    MATCH (:Season {{season_name:'{season}'}})-[:HAS_GW]->(g)-[:HAS_FIXTURE]->(f)
    MATCH (p)-[pi:PLAYED_IN]->(f)
    RETURN p.player_name AS player, SUM(pi.{stat}) AS total_{stat}, pos.name AS position
    ORDER BY total_{stat} DESC LIMIT {limit}
    """
```

```
if intent == "best_players_by_metric" and entities.get("filter_value")!= None:
```

```
    val = entities["filter_value"]
```

```
    return f"""
```

```
        MATCH (:Season {{season_name:'{season}'}})-[:HAS_GW]->(g)-[:HAS_FIXTURE]->(f)
```

```
        MATCH (p:Player)-[pi:PLAYED_IN]->(f)
```

```
        WITH p, SUM(pi.{stat}) AS total_stat
```

```
        WHERE total_stat > {val}
```

```
        RETURN p.player_name AS player, total_stat
```

```
        ORDER BY total_stat DESC LIMIT {limit}
```

```
    """
```

```
if intent == "fixture_details" and player1 and team1:
```

```
    return f"""
```

```
        MATCH (:Season {{season_name:'{season}'}})-[:HAS_GW]->(g)-[:HAS_FIXTURE]->(f:Fixture)
```

```
        MATCH (p:Player {{player_name:'{player1}'}})-[pi:PLAYED_IN]->(f)
```

```
        MATCH (f)-[:HAS_HOME_TEAM]->(h:Team),
```

```
            (f)-[:HAS_AWAY_TEAM]->(a:Team)
```

```
        MATCH (p)-[:PLAYS_FOR]->(t:Team)
```

```
        WHERE h.name = '{team1}' OR a.name = '{team1}'
```

```
        RETURN p.player_name AS player, t.name AS team, f.fixture_number AS fixture, f.kickoff_time, g.GW_number AS gameweek, h, a
```

```
        ORDER BY f.kickoff_time ASC
```

```
    """
```

```

if intent == "Worst_players_by_metric" and not entities.get("position"):
    if entities.get("filter_value")!= None:
        val = entities["filter_value"]
        return f"""
            MATCH (:Season {{season_name:'{season}'}})-[:HAS_GW]->(g)-[:HAS_FIXTURE]->(f)
            MATCH (p:Player)-[pi:PLAYED_IN]->(f)
            WITH p, SUM(pi.{stat}) AS total_stat
            WHERE total_stat > {val}
            RETURN p.player_name AS player, total_stat
            ORDER BY total_stat ASC LIMIT {limit}
        """
    else:
        return f"""
            MATCH (:Season {{season_name:'{season}'}})-[:HAS_GW]->(g)-[:HAS_FIXTURE]->(f)
            MATCH (p:Player)-[pi:PLAYED_IN]->(f)
            RETURN p.player_name AS player, SUM(pi.{stat}) AS total_{stat}
            ORDER BY total_{stat} ASC LIMIT {limit}
        """

if intent == "Worst_players_by_metric" and entities.get("position"):
    position = entities["position"]
    if entities.get("filter_value")!= None:
        val = entities["filter_value"]
        return f"""
            MATCH (p:Player)-[:PLAYS_AS]->(pos:Position {{name:'{position}'}})
            MATCH (:Season {{season_name:'{season}'}})-[:HAS_GW]->(g)-[:HAS_FIXTURE]->(f)
            MATCH (p)-[pi:PLAYED_IN]->(f)
            WITH p, pos, SUM(pi.{stat}) AS total_stat
            WHERE total_stat > {val}
            RETURN p.player_name AS player, total_stat, pos.name AS position
            ORDER BY total_stat ASC LIMIT {limit}
        """

```

#continuation of past intent query formation

else:

```
    return f"""
```

```
        MATCH (p:Player)-[:PLAYS_AS]->(pos:Position {{name:'{position}'}})
```

```
        MATCH (:Season {{season_name:'{season}'}})-[:HAS_GW]->(g)-[:HAS_FIXTURE]->(f)
```

```
        MATCH (p)-[pi:PLAYED_IN]->(f)
```

```
        RETURN p.player_name AS player, SUM(pi.{stat}) AS total_{stat}, pos.name AS position
```

```
        ORDER BY total_{stat} ASC LIMIT {limit}
```

```
    """
```

if intent == "Worst_players_by_metric" and entities.get("filter_value")!= None:

```
    val = entities["filter_value"]
```

```
    return f"""
```

```
        MATCH (:Season {{season_name:'{season}'}})-[:HAS_GW]->(g)-[:HAS_FIXTURE]->(f)
```

```
        MATCH (p:Player)-[pi:PLAYED_IN]->(f)
```

```
        WITH p, SUM(pi.{stat}) AS total_stat
```

```
        WHERE total_stat > {val}
```

```
        RETURN p.player_name AS player, total_stat
```

```
        ORDER BY total_stat ASC LIMIT {limit}
```

```
    """
```

if intent == "player_information" and player1:

```
    return f"""
```

```
        MATCH (p:Player {{player_name:'{player1}'}})-[:PLAYS_FOR]->(t:Team),
```

```
        (p)-[:PLAYS_AS]->(pos:Position)
```

```
        RETURN p.player_name AS player,
```

```
        t.name AS team,
```

```
        pos.name AS position
```

```
    """
```

```
return f"""MATCH (n:NonExistentLabel)
```

```
    RETURN n
```

```
"""
```

Query Result Formatting

Convert raw database results into user-friendly responses.

Handle empty results with clear messages.

Format output based on intent (fixtures, performance, best/worst players, player info).

Include relevant details: player/team, stats, gameweek, and season.

Prompt: When does Harry Kane play against Liverpool?

Query Result:

```
[{'player': 'Harry Kane', 'team': 'Spurs', 'fixture': 148, 'f.kickoff_time': '2022-11-06 16:30:00+00:00', 'gameweek': 15, 'h': {'name': 'Spurs'}, 'a': {'name': 'Liverpool'}}, {'player': 'Harry Kane', 'team': 'Spurs', 'fixture': 338, 'f.kickoff_time': '2023-04-30 15:30:00+00:00', 'gameweek': 34, 'h': {'name': 'Liverpool'}, 'a': {'name': 'Spurs'}}]
```

Formatted Response:

Upcoming Fixtures:

1. Harry Kane in team Spurs will play against Liverpool in Gameweek 15 at 2022-11-06 16:30:00+00:00 (Fixture #148)
2. Harry Kane in team Spurs will play against Liverpool in Gameweek 34 at 2023-04-30 15:30:00+00:00 (Fixture #338)

Embeddings Approach Selected: Feature Vector Embeddings

We focused on generating feature vector embeddings for each row in the CSV. This captures the numerical information of a player's performance in a specific fixture, by storing all the stats of the fixture into one string represented by `feature_text` which was then stored on the `PLAYED_IN` relationship along with the other fixture stats. Which was better than older approach of summarizing player performance through whole season into one sentence.

Example:

Player ID: 72, Player Name: Aaron Connolly, Season: 2021-22, Team: Brighton, Gameweek: 1, Position: FWD, Fixture: 2, Home Team: Burnley, Away Team: Brighton, Kickoff Time: 2021-08-14 14:00:00+00:00, Total Points This Gameweek: 0, Goals Scored This Gameweek: 0, Assists This Gameweek: 0, Minutes Played This Gameweek: 0, Bonus Points This Gameweek: 0, Clean Sheets This Gameweek: 0, Yellow Cards This Gameweek: 0, Red Cards This Gameweek: 0, Own Goals This Gameweek: 0, Penalties Saved This Gameweek: 0, Penalties Missed This Gameweek: 0, Saves: 0, Form Score: 0.0, Value: 55, bonus point system: 0, ict_index: 0.0, Influence: 0.0, Creativity: 0.0, Threat: 0, Total Points So Far: 0, Goals Scored So Far: 0, Assists So Far: 0, Minutes Played So Far: 0, Bonus Points So Far: 0, Clean Sheets So Far: 0

Embedding Models

Three sentence embedding models were evaluated for their balance of performance and efficiency in generating embeddings for textual features:

sentence-transformers/all-
MiniLM-L6-v2

(MiniLM)

sentence-transformers/all-
mpnet-base-v2

(MPNet)

BAAI/bge-base-en-v1.5

(BGE)

Embeddings for each row were generated using all 3 models and stored.

Similarity search was used to retrieve similar results to the input prompt with the chosen model.

Embedding Model Comparison & Results

Speed Test (60 queries)

MiniLM: 0.1443 seconds

MPNet: 0.2810 seconds

BQE: 0.0452 seconds

Key Finding: BQE is 3.19x faster than MiniLM and 6.22x faster than MPNet

Quality Comparison (Similarity Score)

Model	Similarity Score
MiniLM	0.7279
MPNet	0.8037
BQE	0.8504

Query: How did Salah perform against Spurs?

BAAI/bge-base-en-v1.5 (BQE) emerges as the optimal choice, delivering superior semantic relevance (0.9193 similarity score) while maintaining exceptional speed performance.

Embedding Model Comparison: Human-Eye Testing

QUERY: How did Salah perform against Spurs?

MiniLM	Player ID: 487, Player Name: Bryan Gil Salvatierra, Season: 2021-22, Team: Spurs, Gameweek: 29, Position: MID, Fixture: 153, Home Team: Brighton, Away Team: Spurs, Kickoff Time: 2022-03-16 19:30:00+00:00, Total Points This Gameweek: 0, ...
MPNet	Player ID: 233, Player Name: Mohamed Salah, Season: 2021-22, Team: Liverpool, Gameweek: 31, Position: MID, Fixture: 306, Home Team: Liverpool, Away Team: Watford, Kickoff Time: 2022-04-02 11:30:00+00:00, Total Points This Gameweek: 3, ...
BGE	Player ID: 233, Player Name: Mohamed Salah, Season: 2021-22, Team: Liverpool, Gameweek: 18, Position: MID, Fixture: 177, Home Team: Spurs, Away Team: Liverpool, Kickoff Time: 2021-12-19 16:30:00+00:00, Total Points This Gameweek: 2, ...

LLM Layer: Context Construction & Prompt Engineering

The LLM layer is responsible for synthesising the retrieved information into a coherent and accurate answer. Effective context construction and prompt engineering are paramount for optimal LLM performance.

1

Context Construction

We integrate outputs from both the baseline Cypher retrieval and the embedding-based search. This combined context provides the LLM with a rich, multi-faceted understanding of the user's query and relevant FPL data. The structure is:

```
Cypher Results:\n{formatted_cypher}\n\n
```

```
Embedding Results:\n{embedding_context}
```

2

Prompt Structure

Our prompt engineering focuses on explicitly defining the LLM's role and guiding its response. The prompt instructs the LLM to act as an "expert Fantasy Premier League assistant" and strictly use the provided context to answer the user's question, preventing hallucination.

You are an expert Fantasy Premier League assistant. Use the context below to answer the user's question.

The context contains two parts of information: results retrieved from the Neo4j knowledge graph using Cypher queries, and relevant text snippets retrieved using vector embeddings.

Use whichever information is relevant to answer the question, perhaps combining both for a comprehensive response.

However, give the cypher results higher priority as they are more structured and accurate.

Do not state where the information came from in your answer.

```
<context> {context} </context>
```

```
Question: {input}
```



LLM Comparison: Quantitative & Qualitative Evaluation



Gemma-2-2B



Llama-3.2-3B



Mistral-7B

LLM Comparison: Quantitative & Qualitative Evaluation



Response Time (s)

Average time taken for the LLM to generate a complete response.



Tokens (Total)

The sum of input and output tokens processed per query, impacting cost and speed.



Accuracy

The degree to which the LLM's response provides factually correct information.



Relevance

How pertinent the LLM's answer is to the user's specific question or intent.



Completeness

The extent to which the response covers all necessary aspects of the query.



Naturalness

How well the LLM's language flows and sounds like human-generated text.



Correctness

Overall factual and grammatical integrity of the generated response.



Avg Quality

An aggregate score reflecting the overall user experience and utility of the response.

LLM Performance Metrics: Detailed Comparison

Evaluation Comparison Results:

Model	Response Time (s)	Tokens (Total)	Accuracy (1-5)	Relevance (1-5)	Completeness (1-5)	Naturalness (1-5)	Correctness (1-5)	Avg Quality (1-5)
Gemma-2-2B	10.20	2141.75	5.00	5.00	4.50	5.0	5.0	4.9
Llama-3.2-3B	10.42	1929.25	4.25	4.50	4.25	5.0	4.0	4.4
Mistral-7B	12.83	2427.75	4.00	4.75	4.25	5.0	3.5	4.3

The evaluation clearly indicates **Gemma-2-2B** as the top performer, excelling in overall quality and speed. While **Llama-3.2-3B** demonstrates competitive token efficiency, Gemma-2-2B strikes the best balance across all critical metrics for our application.

Remaining Limitations

Through systematic error analysis, we identified key areas for improvement to enhance the reliability and user experience of our FPL AI Assistant.



Ambiguous Entity Resolution

Sometimes, player names or team abbreviations can be ambiguous. Future improvements include implementing more sophisticated disambiguation techniques, possibly leveraging context from previous turns in a conversation or a confidence score during entity extraction.



Complex Query Handling

Queries involving multiple conditions or temporal reasoning (e.g., "best player over the last five gameweeks") occasionally challenged the Cypher query generation. Enhancements will focus on expanding the complexity of Cypher templates and refining intent classification for such cases.



LLM Hallucination Mitigation

While prompt engineering reduced hallucination, it was not entirely eliminated. Exploring advanced RAG techniques, such as re-ranking retrieved documents or fine-tuning LLMs on FPL-specific data, can further minimise this.



User Interface Feedback Loop

Integrating user feedback mechanisms within the UI (e.g., "Was this answer helpful?") can provide valuable data for continuous model training and system refinement.

Live Demonstration: FPL AI Assistant in Action

Join us for a live demonstration of the complete FPL AI Assistant pipeline, from raw input to a generated, data-driven answer.

End-to-End Pipeline Walkthrough

We will showcase the seamless integration of each component, illustrating how a natural language query is transformed into a precise FPL insight.

Dynamic Model Switching

Observe the system's flexibility as we toggle between different embedding models (MiniLM, MPNet, BGE) and LLMs (Gemma, Llama, Mistral) to demonstrate their impact on response quality.

Interactive Q&A Session

We will answer your chosen FPL questions live, demonstrating the assistant's ability to handle diverse queries and provide accurate, contextually relevant information.

Integrated UI Reflecting Process

The user interface has been designed to visually represent the background processes, offering transparency into how the answer is derived.

Thank You
Any Questions?

