Politecnico di Torino

Master's Degree in Mathematical Engineering
Academic Year 2024/2025

Natural Language Generation for Automated Sports Broadcasting

Supervisor:
Riccardo COPPOLA
Anna ARNAUDO

Candidate:
Davide OMENTO

Abstract

This thesis explores the design and implementation of an intelligent system based on a Retrieval-Augmented Generation (RAG) architecture integrated with a Large Language Model (LLM), aimed at automatically generating football match commentary from minimal input data. The system combines neural text generation with targeted retrieval of player and team statistics, enabling the production of detailed and contextually relevant narratives.

The approach encompasses the development of an event annotation framework and a user-friendly interface that enables users to select match events and provide relevant contextual details. A thoughtfully engineered prompting strategy, complemented by few-shot examples, directs the LLM to generate coherent and contextually precise commentary while maintaining factual integrity, including information such as goal scorer, assist provider, type of shot, and event timing. Evaluation covers both quantitative metrics-such as accuracy and coverage of events-and qualitative measures, including human judgments of clarity, informativeness, and narrative quality.

The results indicate that the RAG-based LLM system achieves an event-level accuracy of 99% and that, based on human evaluations, the generated commentaries were rated at a level comparable to human-written ones. This architecture therefore provides a flexible and scalable solution for automated sports commentary, with potential applications in live broadcasting, online platforms, and as a supportive tool for human commentators.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context and Motivation

Sports have become increasingly important globally, and live football broadcasting has grown into a major industry, attracting millions of passionate viewers. This thesis aims to develop an AI agent capable of generating football commentary from minimal input information, by leveraging player and team statistics to enrich the narration with relevant details.

The proposed system could be valuable for applications and websites that provide live, minute-by-minute commentary, which currently often rely on template-based architectures that simply replace player and team names. With this approach, a user can select a match event and provide a few specific details; the AI then tries to generate a complete, natural-sounding commentary.

Furthermore, this work could support smaller football organizations, such as lower-division teams that are not professionally broadcasted but have a big supporter base. By generating live commentary, fans can follow the game from home, even when the team is playing away and no live stream is available.

1.2 Main Contribution

This thesis makes several key contributions to the study and application of Large Language Models (LLMs) for generating football commentaries. The main contributions can be summarized as follows:

- **Theoretical Study of LLMs:** Provided a deep analysis of the principles and architectures of Large Language Models, focusing on their capabilities, limitations, and suitability for narrative generation in sports broadcasting.
- **State-of-the-Art Survey:** Reviewed existing research on automated commentary and LLM applications in broadcasting, identifying gaps in expressiveness, factual grounding, and event-specific adaptability.
- **Structured Prompt Design:** Developed specific prompts to guide LLMs in producing vivid, accurate, and context-aware football commentary, minimizing hallucinations while maintaining narrative richness.
- **Event Annotation and Context Integration:** Designed an annotation framework that categorizes match events and incorporates player and match statistics, enabling richer and more informative commentary.
- **Quantitative and Qualitative Evaluation:** Generated 200 commentary samples, with only two instances of hallucinated information, demonstrating high accuracy. Human evaluation shows that generated commentaries achieve near-human levels in clarity, coherence, and structured coverage of events.

Collectively, these contributions advance both the theoretical understanding and practical application of LLMs in sports commentary, providing a foundation for more expressive contextually grounded automated narration in the future.

1.3 Thesis Structure

This thesis is organized into six main chapters, each addressing a specific aspect of the research on generating football commentaries with Large Language Models:

- **Chapter 1: Introduction** defines the background and motivation for this work, outlines the research objectives, and highlights the main contributions of the thesis.
- **Chapter 2: Theoretical Foundations** provides the theoretical background on Large Language Models, including their architectures, capabilities, and limitations, as well as an overview of existing research on automated sports commentary.
- **Chapter 3: Methodology** presents the approach adopted in this study, detailing the processes of data collection and preprocessing, the design of event representations, and the commentary generation pipeline. Particular emphasis is placed on the structured prompt design and the Retrieval-Augmented Generation architecture built on API-driven data.
- **Chapter 4: Results** presents the experimental evaluation, including both quantitative and qualitative analysis, and discusses the performance of the proposed system in terms of accuracy, coherence, and expressiveness.
- **Chapter 5: Interactive Application Design** illustrates the implementation of the interface that allows users to select events, provide minimal input, and generate commentary automatically, highlighting usability and practical considerations.
- **Chapter 6: Conclusion** summarizes the main findings and contributions of the thesis, discusses limitations, and outlines directions for future work in the area of automated sports commentary.

This structure provides a logical progression from theoretical background to practical implementation and evaluation, guiding the reader through both the conceptual and applied aspects of the research.

Chapter 2

Theoretical Foundations

2.1 Language Models

Language Models (LMs) are a core component of Natural Language Processing (NLP), as they estimate the probability of word sequences and enable coherent text generation [14]. Early approaches relied on statistical methods, such as *n-grams*, which were limited in capturing long-range dependencies.

Neural networks, including Recurrent Neural Networks (RNNs) and Long Short-Term Memory networks (LSTMs), improved sequence modeling by maintaining context across longer texts [32, 52]. However, a major breakthrough came with the Transformer architecture [80], which uses self-attention mechanisms to efficiently capture dependencies across entire sequences.

Modern Large Language Models, such as GPT, BERT, and T5 [16, 23, 60, 62], are pretrained on massive corpora of data and can be used to perform a wide variety of tasks. They have shown great capabilities in text generation, summarization, and question answering. Despite their power, some challenges remain, such as bias, hallucination and high computational costs.

This section provides a detailed overview of the theory behind Language Models, from classical statistical approaches to modern Large Language Models. It introduces the mathematical foundations, architectures, and training objectives used in NLP.

2.1.1 Statistical Language Models

A language model assigns a probability to a sequence of words, allowing the prediction of the next word based on the previous context. Statistical language models estimate the probability of a word sequence by modeling the conditional probability of each word given its preceding words. For a sequence of T words $w_1, w_2, \ldots, w_T$, the probability of the entire sequence can be expressed as [14]:

$$\hat{P}(w_1^T) = \prod_{t=1}^{T} \hat{P}(w_t \mid w_1^{t-1}),$$

where $w_i^j = (w_i, w_{i+1}, \ldots, w_j)$ is the sub-sequence from word i to j.

To reduce modeling complexity and data sparsity, *n-gram* models approximate the conditional probability of the next word using only the previous $n-1$ words [14]:

$$\hat{P}(w_t \mid w_1^{t-1}) \approx \hat{P}(w_t \mid w_{t-n+1}^{t-1}).$$

While *n-gram* models effectively capture short-range dependencies, they suffer from data sparsity, cannot model long-range dependencies, and become computationally expensive for large vocabularies or higher orders. These limitations motivated the development of neural language models, which use distributed word representations to capture semantic and syntactic patterns and allow modeling of longer contexts.

2.1.2 RNN and LSTM Based Models

Recurrent Neural Networks for Language Modeling The use of neural networks for language modeling was first introduced by Bengio et al. [14] with feedforward networks using fixed-length context windows. Although effective, these models cannot capture long-term dependencies due to the fixed context size. It was demonstrated [29,71] that neural network language models outperform traditional statistical models, particularly in speech recognition tasks.

To overcome the fixed-context limitation, Mikolov et al. [52] proposed Recurrent Neural Networks, which maintain a dynamic hidden state $h(t)$ that allows the model to encode arbitrarily long contexts:

$$h(t) = f\left(Ux(t) + Wh(t-1)\right), \quad y(t) = g\left(Vh(t)\right),$$

where $x(t)$ is the current word input, $h(t)$ the hidden state, and $y(t)$ the output probability distribution over the vocabulary. The activation function $f(\cdot)$ is typically a sigmoid, while $g(\cdot)$ is a softmax which needs to obtain valid probabilities.

Training uses stochastic gradient descent with backpropagation through time (BPTT), minimizing the cross-entropy loss, which is the objective function, between the predicted distribution $y(t)$ and the true next word. Truncated BPTT propagates error signals for a limited number of time steps, balancing computational efficiency and long-range dependency capture.

Compared to feedforward models, RNNs require tuning only the hidden layer size and achieve significant perplexity reductions over *n-gram* and feedforward models. Additionally, dynamic RNNs can continue adapting during testing, improving performance in domain-specific scenarios such as speech recognition.

In summary, RNNs address the fixed-context limitation of feedforward neural networks, offering a flexible and powerful framework for language modeling that was state-of-the-art at their introduction. Figure 2.1 illustrates the structure of a RNN with a hidden state.

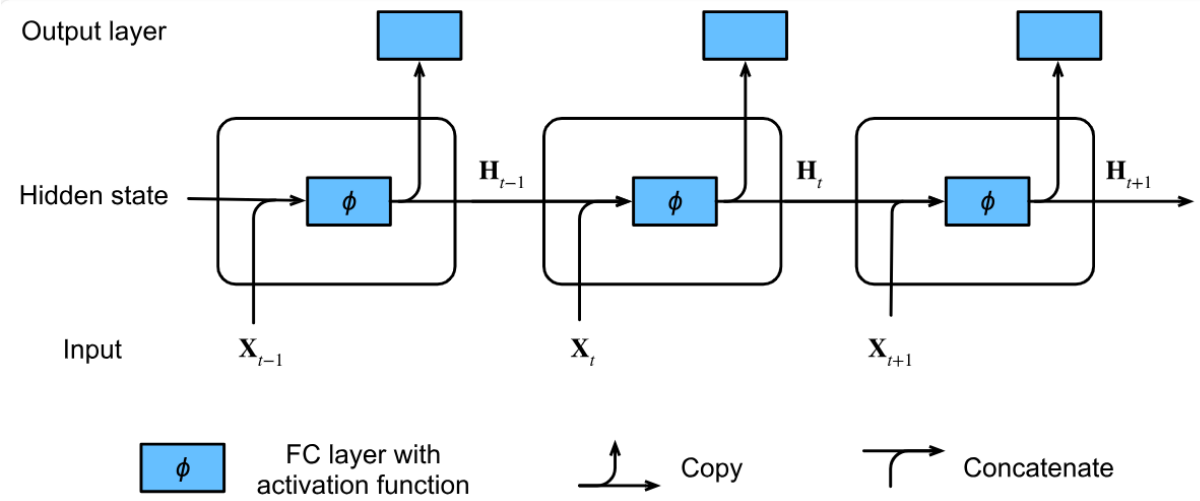

Figure 2.1: Recurrent Neural Network with a hidden state. Source [93].

Long Short-Term Memory Networks Despite their advantages, standard RNNs struggle with learning long-term dependencies due to vanishing or exploding gradients during training. To address this, Hochreiter and Schmidhuber [32] introduced the Long Short-Term Memory (LSTM) network, which incorporates memory cells and gating mechanisms to control information flow.

An LSTM cell contains three gates: input, forget, and output, which regulate how information is added, retained, or output at each time step:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i), \quad f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f),$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o), \quad \tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c),$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \quad h_t = o_t \odot \tanh(c_t),$$

where c_t is the cell state, h_t the hidden state, x_t the input, $\sigma(\cdot)$ the sigmoid activation, and $\odot$ denotes element-wise multiplication. This structure, showed in Figure 2.2, enables the LSTM to mantain relevant information over long sequences and reduce the vanishing gradient problem.

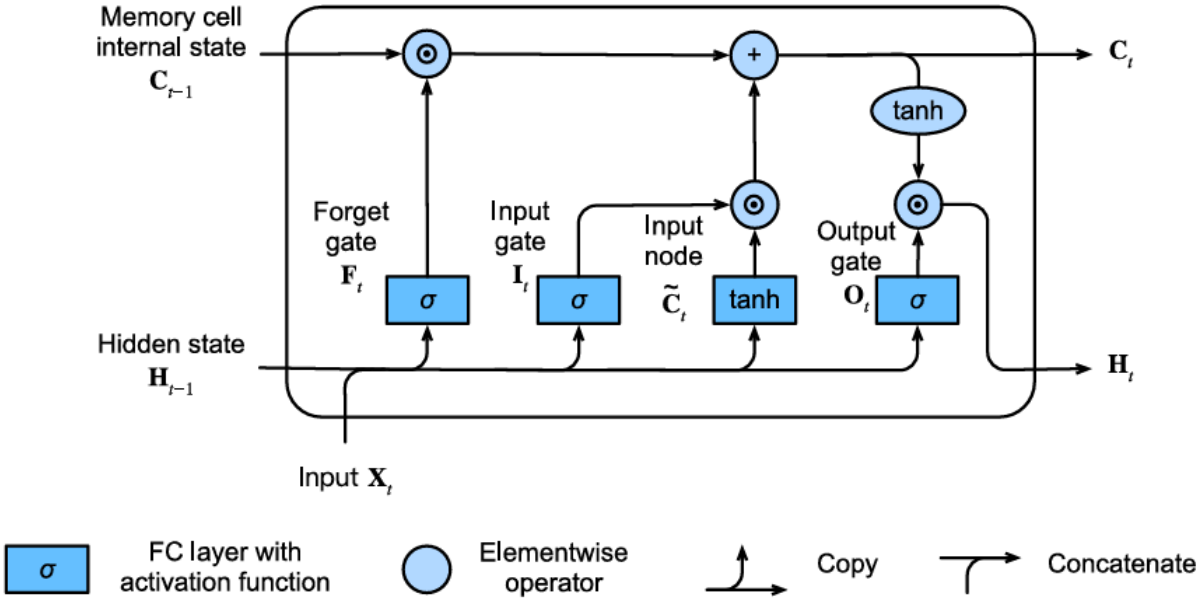

Figure 2.2: Long Short-Term Memory cell architecture. Source: [93].

LSTMs have become a standard in sequence modeling, significantly improving language modeling, machine translation and speech recognition tasks. The combination of the flexibility of RNNs with the robust memory mechanism of LSTMs, allows to obtain a model that can capture both short-term and long-term dependencies in text, offering a more powerful framework for neural language modeling.

Despite their strengths, both RNNs and LSTMs have limitations:

- Computational inefficiency: sequential processing prevents parallelization across time steps, making training slow on long sequences.
- Difficulty in capturing very long dependencies: while LSTMs mitigate vanishing gradients, extremely long contexts can still be challenging to model.
- High memory requirements: maintaining hidden states and gating mechanisms for long sequences can consume substantial memory.

2.1.3 Transformers and Self-Attention

The seminal work *Attention Is All You Need* by Vaswani et al. [80] introduced the Transformer architecture, which fundamentally changed sequence modeling in natural language processing. Unlike previous models based on recurrent or convolutional networks, the Transformer relies entirely on self-attention mechanisms to capture dependencies between sequence elements, enabling greater parallelization and efficiency.

Scaled Dot-Product Attention The core component of the Transformer is the scaled dot-product attention, which computes the relevance of each word in a sequence relative to all others (Figure 2.3). Given queries $Q \in \mathbb{R}^{n \times d_k}$, keys $K \in \mathbb{R}^{n \times d_k}$, and values $V \in \mathbb{R}^{n \times d_v}$, the attention output is:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V,$$

where d_k is the dimensionality of the key vectors. The division by $\sqrt{d_k}$ avoids too large dot-products, stabilizing gradients. This allows the model to dynamically weigh the importance of different tokens, effectively capturing short- and long-range dependencies.

Multi-Head Attention Instead of computing a single attention distribution, the Transformer uses multi-head attention (Figure 2.3). The input embeddings are projected into h different subspaces, producing head-specific queries, keys and values:

$$Q_i = XW_i^Q, \quad K_i = XW_i^K, \quad V_i = XW_i^V, \quad i = 1, \ldots, h.$$

Each head computes:

$$\text{head}_i = \text{Attention}(Q_i, K_i, V_i).$$

The results of all heads are concatenated and linearly projected:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O.$$

This design allows different heads to focus on different types of relationships simultaneously, for example syntactic and semantic, yielding richer contextualized representations.

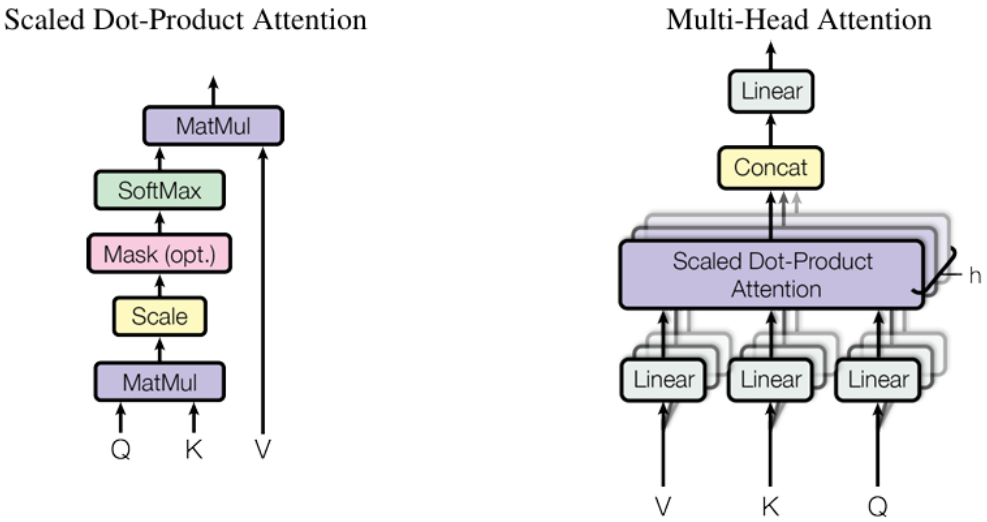

Figure 2.3: Left: scaled dot-product attention. Right: multi-head attention. Source: [80].

Encoder and Decoder The encoder is composed of N identical layers, each with two sub-components:

1. Multi-head self-attention: each token attends to all tokens in the input sequence, updating its representation contextually. By employing multiple attention heads, the model can capture different types of dependencies (e.g., syntactic and semantic relations) in parallel.

2. Feedforward network (FFN): a two-layer multi-layer perceptron (MLP) with ReLU or GELU activation is applied independently to each position, thereby introducing non-linearity and enhancing the representational capacity of the network.

Each sublayer is wrapped with residual connections and layer normalization:

$$\text{LayerNorm}(X + \text{Sublayer}(X)),$$

which helps stabilize training, mitigate vanishing gradients, and accelerate convergence.

By stacking N layers, the encoder is able to progressively refine token embeddings, yielding deep contextual representations of the input sequence that are sensitive to both local and long-range dependencies. This hierarchical structure makes the encoder particularly effective in capturing complex patterns in natural language.
The decoder is also composed of N layers, each containing three sub-components:

1. Masked multi-head self-attention: prevents attending to future tokens, ensuring autoregressive generation and preserving causality in the sequence.

2. Encoder-decoder attention: each position in the decoder attends to encoder outputs, enabling conditioning on the source sequence and facilitating alignment between input and output.

3. Feedforward network: as in the encoder, a position-wise MLP is applied, enriching the representations after the attention mechanisms.

This architecture allows the decoder to generate tokens sequentially while leveraging both past outputs and encoder information. In practice, this means that the model can combine knowledge of what has already been generated with a global understanding of the input, leading to coherent and contextually accurate outputs.

Positional Encodings Since the Transformer architecture does not rely on recurrence or convolution, it requires a mechanism to encode the order of tokens in a sequence. This is achieved through positional encodings, which are added element-wise to the input embeddings. The most common approach uses sinusoidal functions of different frequencies, defined as follows:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right), \quad PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right).$$

The notation is specified as follows:

- pos denotes the position of the token in the sequence (e.g., $pos = 0$ for the first token, $pos = 1$ for the second, and so on).

- i is the index of the embedding dimension.

- d_{model} is the dimensionality of the embeddings (for example, $d_{model} = 512$ in the original Transformer).

- Even dimensions ($2i$) use a sine function, while odd dimensions ($2i + 1$) use a cosine function. This alternation ensures that each position is encoded by a unique combination of sinusoidal values.

- The denominator $10000^{2i/d_{model}}$ acts as a scaling factor, producing sinusoidal curves with different frequencies across the embedding dimensions. Low-index dimensions vary slowly, while high-index dimensions vary more rapidly.

This design allows the positional encodings to represent both absolute position, since each pos maps to a distinct vector, and relative position, since differences between positions correspond to phase shifts of the sinusoids. Consequently, the Transformer can exploit sequential order information without explicit recurrence or convolutional operations.

Overall Architecture The complete encoder-decoder architecture is illustrated in Figure 2.4. The encoder produces contextualized embeddings, which the decoder autoregressively uses to generate the output sequence.

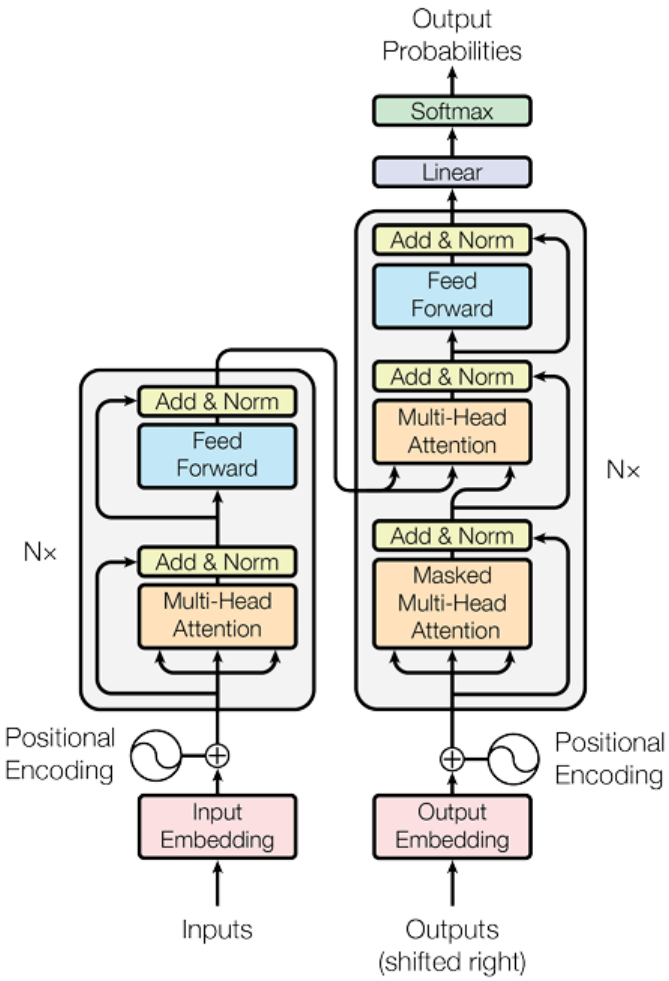

Figure 2.4: Encoder-decoder architecture diagram. Source: [80].

Transformers outperform RNNs and LSTMs by efficiently capturing long-range dependencies while enabling highly parallelizable computation, significantly reducing training time on large datasets. They have set new benchmarks in machine translation, language modeling, and text generation [23, 60].

Limitations of Transformers Despite their success, Transformers have some notable limitations:

- Self-attention scales quadratically with sequence length ($O(n^2)$), leading to high memory and computational costs for very long inputs.
- The model relies on positional encodings for order, which may be insufficient to capture hierarchical or structured dependencies.
- Training often requires extremely large datasets and computational resources (GPUs/TPUs).
- Self-attention can sometimes diffuse over irrelevant tokens, reducing efficiency on long or noisy inputs.

2.1.4 Large Language Models

Building on the Transformer architecture described in the previous section, Large Language Models extend its scalability to unprecedented levels. By training models with billions of parameters on massive corpora of text, LLMs are able to capture long-range dependencies, semantics, and complex linguistic structures [16,23,60,62]. This has enabled a shift from specialized architectures for individual tasks to general-purpose models capable of solving a wide variety of problems through the same underlying framework.

Unlike earlier approaches such as Recurrent Neural Networks or statistical language models, LLMs exploit the self-attention mechanism to efficiently encode context across long sequences. This allows them not only to generate fluent and contextually appropriate text, but also to summarize documents, answer factual and reasoning-based questions, translate between languages, and even perform tasks such as code synthesis or multi-turn dialogue.

Pretraining Paradigms and Representative Models The success of LLMs is based on large-scale pretraining, with different families of models adopting distinct objectives:

- Autoregressive language modeling (causal modeling): models such as the GPT series [16, 60] are trained to predict the next token given only the previous ones. This makes them highly effective for open-ended text generation and dialogue. GPT-3 demonstrated strong zero-shot and few-shot learning capabilities, while GPT-4 further improved reasoning ability and alignment with user intent.
- Masked language modeling (bidirectional modeling): BERT [23] and its successors mask a subset of tokens and train the model to reconstruct them using information from both past and future context. This leads to strong bidirectional representations that are particularly useful for discriminative tasks like classification, named entity recognition, and information retrieval.
- Sequence-to-sequence pretraining: models such as T5 [62] cast all NLP tasks into a text-to-text format, enabling a single architecture to generalize across translation, summarization, and question answering. This task-unification perspective has been influential in shaping the design of more recent foundation models.

Adaptation Strategies Once pretrained, LLMs can be adapted to downstream tasks in different ways:

- Full fine-tuning: all parameters are updated on the target dataset. This achieves strong performance but is computationally expensive for large models.
- Parameter-efficient fine-tuning (PEFT): techniques such as adapters, prefix-tuning, or Low-Rank Adaptation (LoRA) update only a small subset of parameters, significantly reducing resource demands while retaining competitive accuracy.
- Prompt-based adaptation: in many applications, LLMs are not fine-tuned at all. Instead, they are guided through carefully designed prompts or through in-context learning, where the model infers the task from a few demonstrations embedded in the input. This has become a defining feature of modern foundation models.

Strengths and Limitations The versatility of LLMs has made them the cornerstone of modern NLP, yet they also come with challenges. Table 2.1 and Table 2.2 show respectively strenghts and limitations of LLMs.

Strength	Description
Generalization	Work across a wide range of tasks without task-specific architectures.
Long-range dependencies	Capture long-range dependencies effectively through self-attention.
Fluent generation	Produce fluent, human-like text that often rivals human-written output.

Table 2.1: Strengths of Large Language Models.

Limitation	Description
Hallucinations	Tend to produce factually incorrect or fabricated statements.
Biases	Reflect and amplify stereotypes present in training data.
Resource intensity	Training and inference require massive computational and energy resources, raising concerns about sustainability and accessibility.
Context window limits	Constrained by a finite number of tokens, which hinders reasoning over long documents or dialogues.

Table 2.2: Limitations of Large Language Models.

Thus LLMs represent the natural evolution of the Transformer architecture, scaling its capacity and versatility to serve as a foundation for modern AI systems. The next section will examine in greater detail the training objectives that characterize these models, and how different pretraining strategies shape their downstream performance.

2.1.5 Training Objectives

Large Language Models are typically trained with the goal of modeling the probability distribution of text. Given a sequence of tokens $w_1, \ldots, w_T$, the training objective maximizes the likelihood of observing the sequence under the model:

$$\mathcal{L} = \sum_{t=1}^{T} \log P(w_t \mid w_1, \ldots, w_{t-1}).$$

This formulation expresses the idea that a model learns to predict each token w_t conditioned on the tokens that precede it. Minimizing the negative of this objective corresponds to minimize the cross-entropy loss, which is the standard choice in practice. Intuitively, pretraining on large corpora allows the model to capture general language regularities, such as grammar, semantics, and discourse structure, before any task-specific fine-tuning.

From this general formulation, different training strategies emerge depending on how the conditional probability $P(w_t \mid \cdot)$ is defined. A first and most direct approach is autoregressive (causal) modeling, as employed by GPT-style models [16, 61]. As already discussed, here the conditioning is strictly left-to-right: the model predicts the next token using only the sequence of previous tokens, without access to future ones:

$$\mathcal{L}_{\text{AR}} = -\sum_{t=1}^{T} \log P(x_t \mid x_1, \ldots, x_{t-1}).$$

This causal setup closely mirrors the generative process of natural language, where each new word follows from the previous ones. As a result, it promotes fluent text generation and supports zero-shot or few-shot generalization when the model is prompted with specific instructions.

In contrast, masked language modeling (MLM), used in BERT [23], masks a subset of tokens and requires the model to reconstruct them from the surrounding context:

$$\mathcal{L}_{\text{MLM}} = -\sum_{t \in \mathcal{M}} \log P(x_t \mid x_{\setminus \mathcal{M}}),$$

where $\mathcal{M}$ is the set of masked positions. By leveraging bidirectional context, MLM fosters rich semantic representations, making it especially effective for discriminative tasks such as classification, retrieval, or entailment, where the sequential generation is not needed.

A third paradigm is the sequence-to-sequence (seq2seq) objective, popularized by models like T5 [62]. In this framework, tasks are text-to-text transformations, where for an input x the model learns to generate an output y:

$$\mathcal{L}_{\text{seq2seq}} = -\sum_{t=1}^{T_y} \log P(y_t \mid y_{<t}, x).$$

This unified formulation enables diverse applications such as translation, summarization, and question answering within a single architecture.

Beyond these core objectives, LLMs often benefit from auxiliary and alignment objectives. For instance, tasks like next sentence prediction or sentence order prediction [23, 39] encourage discourse-level coherence; contrastive learning [26] helps align semantically similar sentences in the embedding space; and reinforcement learning from human feedback (RLHF) [19, 58] aligns model outputs with human values and preferences, improving factuality, safety, and usefulness.

Finally, the choice of training objective has direct implications for optimization and evaluation. Likelihood-based approaches are typically assessed using perplexity or cross-entropy, while specific downstream tasks rely on metrics such as F1 [79], BLEU [59], ROUGE [47], or accuracy. Scaling further introduces practical challenges: large models demand careful tuning of learning rates, batch sizes, and regularization strategies to ensure stable convergence.

In sum, while autoregressive, masked, and seq2seq objectives define the main paradigms of LLM pretraining, auxiliary and alignment techniques refine these models for greater results and alignment with human needs. The interplay between objective design and optimization strategies is thus central to shaping both the generative fluency and discriminative power of modern LLMs.

2.1.6 Fine-Tuning Large Language Models

Fine-tuning is the process of adapting a pre-trained Large Language Model to perform specific tasks or specialize in particular domains. Unlike training a model from scratch, fine-tuning leverages knowledge acquired during pre-training on vast datasets (Figure 2.5), enabling the model to generalize to new tasks using relatively smaller, task-specific datasets [49].

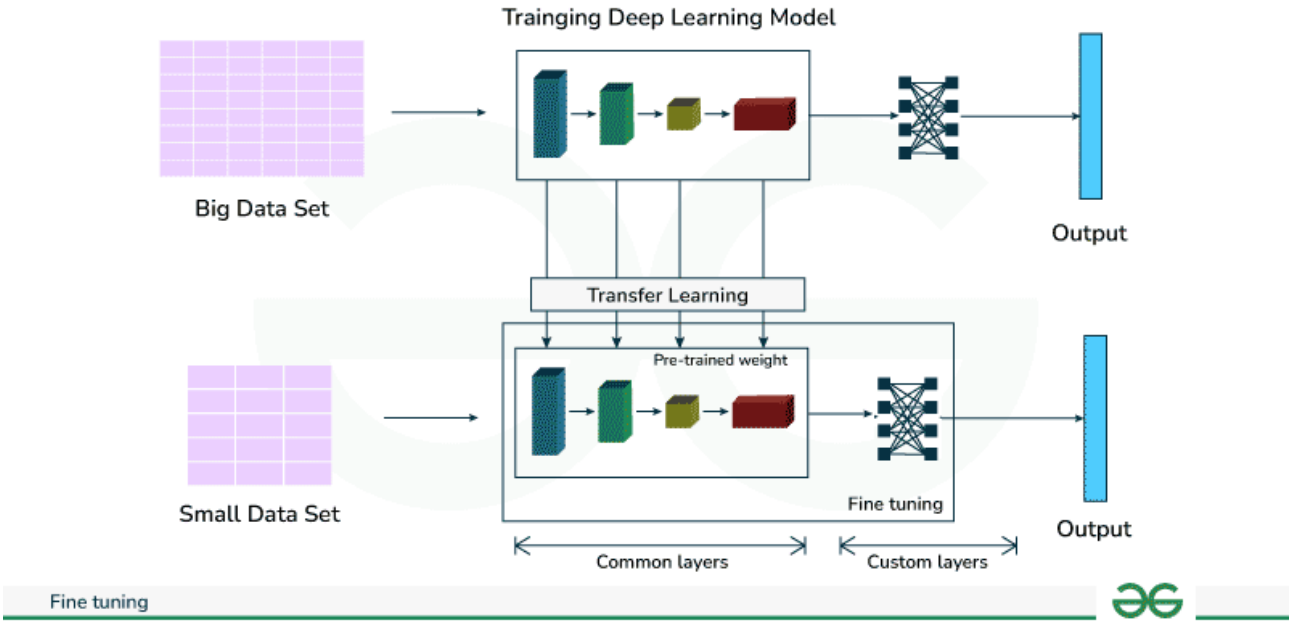

Figure 2.5: Fine-tuning process in deep learning. Source: [28].

Evolution of Fine-Tuning Techniques Early fine-tuning approaches involved updating all parameters of a pre-trained model, which proved to be computationally expensive. To minimize resource

requirements, parameter-efficient fine-tuning techniques have been developed. These methods update only a subset of parameters, maintaining most pre-trained weights reducing complexity [34].

Popular Parameter-efficient fine-tuning (PEFT) strategies include:

- Low-rank adaptation: introduces trainable low-rank matrices into Transformer layers to efficiently adapt the model [34].
- Adapter layers: small networks inserted between existing layers to achieve task-specific adaptations without updating the full model.
- Prompt tuning: learning task-specific prompts to guide the model's behavior without modifying parameters [49].

Recent research has explored techniques such as:

- Memory fine-tuning: helps the model retain knowledge from previous tasks.
- Mixture of experts (MoE): activates subsets of model components per input for efficient specialization.
- Reinforcement learning approaches: methods like PPO and Direct Preference Optimization (DPO) align model outputs with human preferences [49].
- Representation fine-tuning (ReFT): modifies internal representations instead of parameters for lightweight adaptation [49].

Challenges and Best Practices Fine-tuning can introduce issues such as overfitting, negative transfer, or bias amplification. Best practices include:

- Data augmentation and synthetic data generation to generalize better.
- Regularization techniques (dropout, weight decay, early stopping) to reduce overfitting.
- Cross-validation to ensure generalizability.
- Fairness-aware algorithms to mitigate biases [49].

Fine-tuning remains a cornerstone for tailoring LLMs to domain-specific tasks efficiently, balancing performance improvements with computational constraints.

2.1.7 Prompt Engineering

Prompt engineering is the practice of carefully designing input prompts to guide a pre-trained Large Language Model towards generating desired outputs without modifying its underlying parameters. This approach leverages the pre-trained knowledge of LLMs, allowing them to perform complex tasks across domains with minimal additional training. As LLMs have increased in size and capability, prompt engineering has become a critical method to maximize their utility, especially when labeled data is scarce [15, 48].

The field has rapidly expanded, producing taxonomies that organize prompting strategies:

- *The Prompt Report* (Schulhoff et al., 2024) categorizes 58 methods into six themes (reasoning, generation, control, etc.), offering design principles [70].
- Vatsal & Dubey (2024) systematize 39 prompting techniques across 29 NLP tasks, highlighting task-specific adaptability [81].
- Sahoo et al. (2025) provide a systematic survey with emphasis on datasets, application contexts, and evaluation frameworks [68].

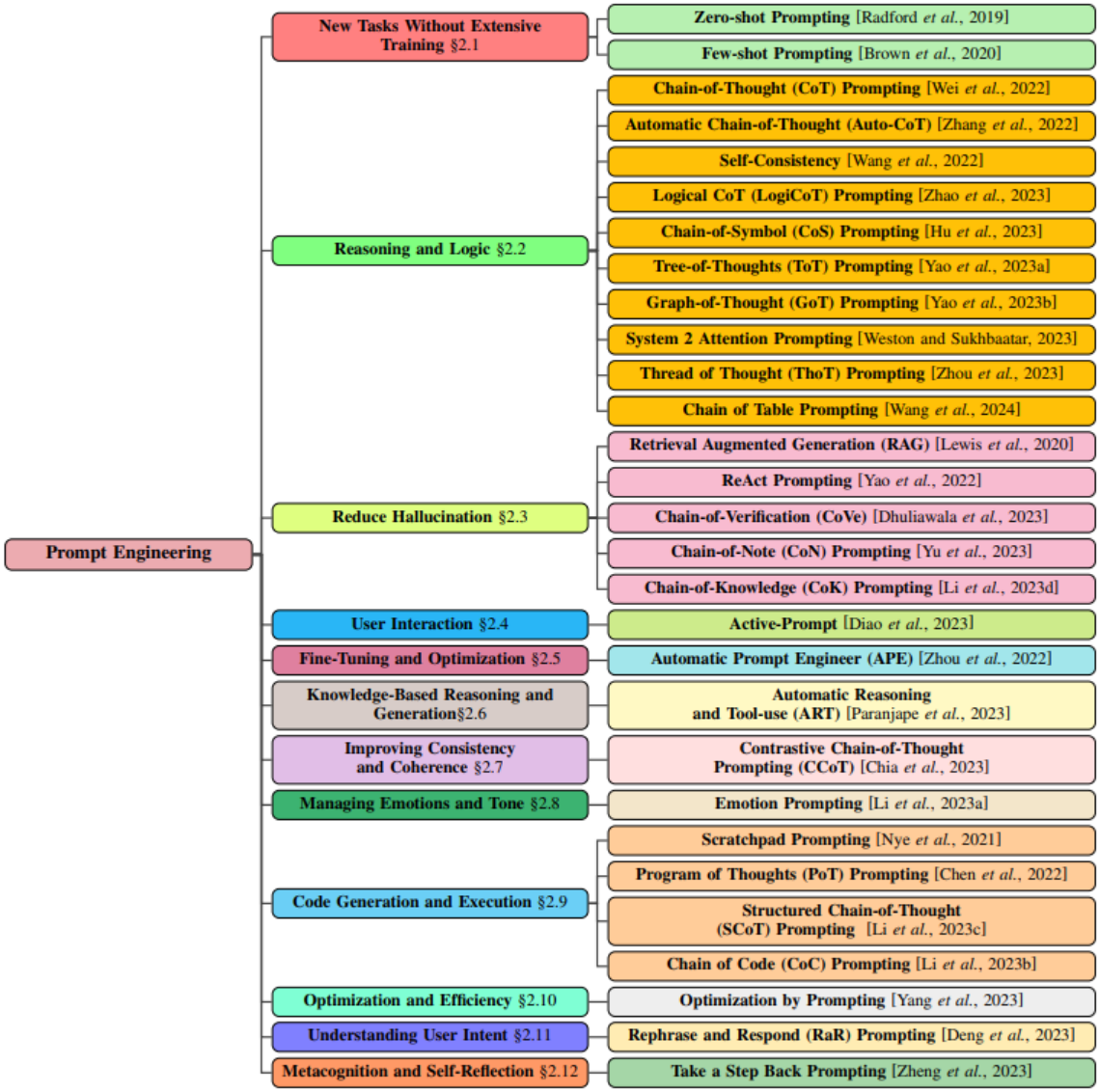

Figure 2.6: Taxonomy of prompt engineering techniques in LLMs, organized around application domains. Source: [68]

These taxonomies serve as practical guides for choosing prompt types based on application constraints. Figure 2.6 illustrates the taxonomy of prompt engineering techniques in LLMs, organized around application domains, providing a nuanced framework for customizing prompts across diverse context [68].
The most common prompting techniques include:

- **Zero-shot and few-shot prompting** (Figure 2.7): *Zero-shot prompting* defines the task purely via simple and natural language instructions, without any exemplars. This relies on emergent generalization capabilities of large-scale LLMs, and its effectiveness grows with model size [16]. However, outputs are often brittle with respect to phrasing [48]. *Few-shot prompting*, introduced in GPT-3 [16], incorporates a handful of labeled input-output pairs inside the prompt. This provides in-context supervision and enables the model to infer task structure dynamically. Few-shot performance strongly correlates with the representativeness of the exemplars, so it's important to include relevant ones [53].

- **Chain-of-Thought (CoT) prompting** (Figure 2.8): CoT prompting elicits intermediate reasoning steps before producing a final answer. This allows LLMs to externalize their latent reasoning, significantly boosting performance on arithmetic (GSM8K), commonsense reasoning

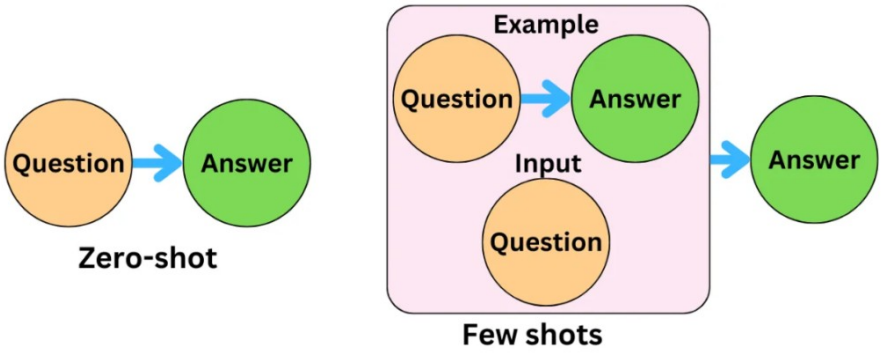

Figure 2.7: Comparison of zero-shot and few-shot prompting approaches. Source: [86].

(StrategyQA), and symbolic tasks [38, 87]. Variants include:

- Few-shot CoT: showing demonstrations with explicit reasoning traces.
- Zero-shot CoT: using trigger phrases like '*Let's think step by step*' [38].

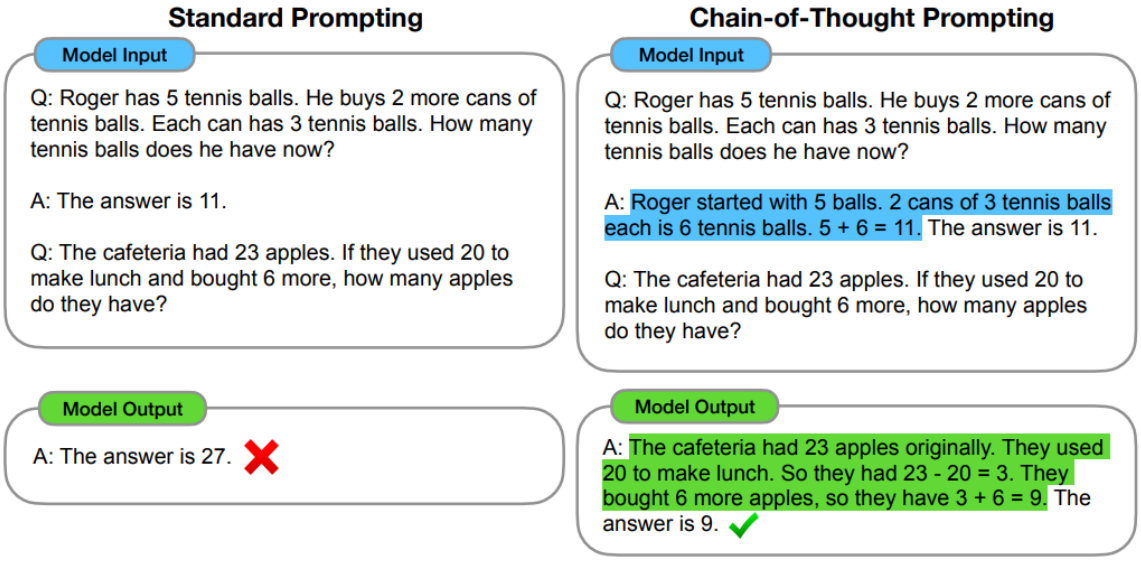

Figure 2.8: Chain-of-thought reasoning processes. Source: [87].

- **Advanced CoT variants**: Beyond vanilla CoT, multiple structured extensions have been proposed:
  - Tabular CoT (Tab-CoT): intermediate reasoning steps are formatted into structured tables, improving clarity and alignment with symbolic reasoning tasks [94].
  - Thread-of-Thought (ThoT): encourages models to break down tasks into conversational or modular substeps, promoting explainability [83].
  - Contrastive CoT: presents both correct and incorrect reasoning chains, enabling models to learn discriminative reasoning patterns [22]. These variants emphasize not only accuracy but also interpretability and robustness.
- **Self-consistency and ensemble prompting**: Self-consistency, introduced by Wang et al. (2022) [84], samples multiple CoT responses under stochastic decoding and aggregates via majority voting. This improves reliability by smoothing variance across reasoning paths. Ensemble prompting generalizes this idea: multiple differently phrased prompts are issued, and responses are aggregated. This reduces sensitivity to single-prompt wording and improves robustness across domains, at the cost of introducing complexity [92].

- **Retrieval-Augmented Generation (RAG)**: RAG integrates prompting with retrieval mechanisms: an external retriever selects relevant knowledge (documents, passages), which is then injected into the prompt context [27,43]. This grounds LLM responses in verifiable evidence and mitigates hallucinations. Prompt engineering in RAG often focuses on how retrieved content is formatted (e.g., inline concatenation vs. structured templates) and how context is ranked or filtered before injection. RAG has proven particularly effective in question answering, domain-specific tasks, and real-time applications.

Despite its significant potential, prompt engineering faces several non-trivial challenges:

- **Sensitivity to wording**: LLM outputs can vary dramatically with minor rephrasing of prompts, even when the underlying task remains the same. For instance, slight changes in phrasing can affect reasoning chains or numerical outputs in arithmetic tasks [48, 87]. This sensitivity complicates prompt design for large-scale deployment, especially in multi-lingual or multi-domain scenarios. Lots of tries are required to reach the best possible prompt.
- **Scalability and manual effort**: Designing high-quality prompts for diverse tasks often requires extensive human expertise. Few-shot or chain-of-thought prompts, in particular, demand carefully curated exemplars [16,69]. Automating prompt generation or optimization (e.g., via prompt tuning or reinforcement learning from human feedback) is still an active research area [42, 95].
- **Hallucinations and factual errors**: Even with carefully crafted prompts, LLMs can generate outputs that are syntactically plausible but factually incorrect. This issue is especially prominent in tasks requiring domain-specific knowledge or real-time information, highlighting the need for retrieval-augmented generation or external grounding [27, 84].
- **Evaluation and benchmarking difficulty**: Measuring the effectiveness of prompts across heterogeneous tasks is non-trivial. Standard metrics such as accuracy or BLEU [59] scores may not capture reasoning quality, creativity, or factual consistency. Recent works propose task-specific benchmarks (e.g., GSM8K for arithmetic reasoning, MATH for formal reasoning) and self-consistency evaluation to provide more reliable assessments [20, 87].
- **Transferability across models and domains**: A prompt effective for one LLM may fail on another due to differences in pretraining corpora or architecture. Cross-model prompt robustness remains limited, and scaling strategies like instruction-tuning or multi-task prompts are required to improve generalization [54, 69].

To mitigate these challenges, effective prompt engineering often combines multiple strategies:

- Iterative prompt refinement: start with an initial prompt, evaluate outputs on task-specific benchmarks, and iteratively adjust wording, examples, or instructions to optimize performance [15, 48].
- Combining few-shot and chain-of-thought prompting: embedding carefully selected exemplars that include reasoning steps significantly improves performance on multi-step tasks, such as arithmetic, commonsense reasoning, or logical inference [38, 87].
- Dynamic and contextual prompting: incorporate external knowledge dynamically via retrieval-augmented generation or domain-specific context injection to reduce hallucinations and enhance factual accuracy [27, 43].
- Automation and optimization: use techniques like prompt tuning, soft prompts, or reinforcement learning from human feedback (RLHF) to reduce manual effort and scale across tasks and models [42, 95].
- Robust evaluation: apply multiple metrics, including accuracy, consistency, and reasoning quality, to ensure the prompt generalizes across task variants and avoids overfitting to specific benchmarks [20, 84].

2.1.8 Retrieval-Augmented Generation

Retrieval-Augmented Generation architectures are typically structured into three interconnected stages: Retrieval, Augmentation, and Generation. These modules work in synergy to supplement the LLM's pretrained internal knowledge with external, task-relevant information [27] and to limit hallucinations.

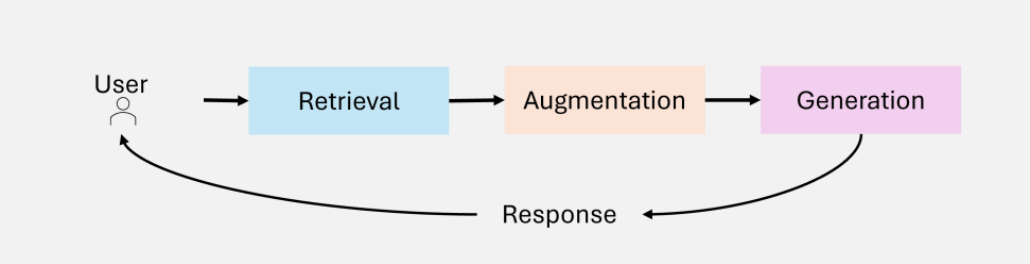

Figure 2.9: Core components of RAG. Source: [72]

- **Retrieval**: This component identifies relevant knowledge units, such as documents, passages, or structured data, from external repositories. Modern implementations rely on dense vector similarity search, traditional IR techniques, or hybrid approaches. Vector databases are frequently employed to efficiently retrieve the top-k most relevant chunks with respect to the encoded query [77].

- **Augmentation**: Retrieved content is integrated into the LLM's context window, either through direct concatenation, summarized text, tabular representations, or key-value pairs. This step ensures that the model grounds its reasoning in domain-specific, timely, or verifiable external knowledge. To improve robustness, augmentation can also include reranking of retrieved passages or query expansion strategies [44].

- **Generation**: In the final stage, the LLM produces an answer conditioned on both the user input and the retrieved context. This hybridization reduces hallucinations and increases factuality by anchoring the generative process to external evidence while leveraging the model's reasoning and linguistic fluency [27].

The **retrieval** module is the first and critical stage in Retrieval-Augmented Generation, responsible for identifying external knowledge relevant to a user query. Its main goal is to supply the generative model with grounded information while minimizing irrelevant or noisy content.

- Query and document encoding: queries and candidate documents are converted into vector embeddings using pretrained models such as BERT, Sentence-BERT, or domain-specific embeddings. This semantic representation allows retrieval based on meaning rather than just keyword matching [64].

- Similarity search: retrieved vectors are compared with the query embedding to find the most relevant items. Dense retrieval methods or sparse methods can be used. Hybrid approaches combining both often provide higher precision and coverage [27].

- Document segmentation: documents are typically split into passages or chunks to improve retrieval granularity. Common approaches include fixed-length windows, sliding windows with overlap, or semantic segmentation. Proper chunking ensures contextually coherent input without exceeding the model's context window [77].

- Ranking and filtering: retrieved candidates are ranked based on relevance scores or metadata. Re-ranking can use lightweight neural models or heuristics to ensure high-quality passages are passed to the generative module [44].

- Key considerations: challenges include ambiguous queries, outdated or noisy data, and propagation of retrieval errors into generation. Effective retrieval design balances recall and precision while ensuring that the generative model receives useful, concise information.

By providing semantically relevant, high-quality passages, the retrieval module lays the foundation for accurate and grounded generation in RAG systems, reducing hallucinations and enhancing factuality.

The **augmentation** module is responsible for integrating retrieved knowledge into the generative model's input in a way that maximizes factual grounding and task relevance. Unlike retrieval, which identifies relevant information, augmentation determines how this information is presented to the model.

- Context injection: retrieved passages, documents, or structured data are inserted into the model's input context. Common strategies include direct concatenation, summarization, or conversion into key-value pairs. The goal is to provide the model with concise, coherent, and task-relevant knowledge [44].

- Handling long contexts: since LLMs have limited context windows, augmentation often involves selecting or summarizing the most relevant chunks, discarding redundant or low-value content. Techniques include passage ranking, content pruning, and sliding-window concatenation to fit within token limits.

- Prompt design and structuring: the way retrieved information is presented significantly impacts model performance. Clear formatting, structured templates, and natural language cues help the model interpret and utilize augmented knowledge effectively.

- Re-ranking and filtering: augmentation can also include additional ranking or filtering steps to ensure that only the most relevant or verified information is incorporated. This may involve task-specific heuristics, cross-encoder scoring, or metadata-based prioritization.

- Challenges and considerations: key challenges include maintaining coherence between the retrieved information and the user query, avoiding context overflow, and preventing the model from relying disproportionately on low-quality passages.

By carefully designing the augmentation stage, RAG systems can effectively bridge retrieval and generation, providing the LLM with a grounded, contextually relevant knowledge base that supports accurate and factually informed outputs.

The **generation** module is the final stage in Retrieval-Augmented Generation, where the language model produces outputs conditioned on both the user query and the augmented context. This step leverages the LLM's reasoning and linguistic capabilities while grounding the response in retrieved knowledge.

- Context-conditioned generation: the model generates responses by attending to the combined input of the user query and retrieved passages. Properly structured augmentation ensures that the generated content is factual, relevant, and coherent [27].

- Reduction of hallucinations: by grounding the generation in retrieved knowledge, RAG significantly reduces the likelihood of hallucinations, providing evidence-supported answers. The quality of retrieval and augmentation directly influences factual accuracy [44].

- Iterative and multi-hop reasoning: advanced RAG systems may use iterative generation, where the model queries additional information based on intermediate outputs, or multi-hop reasoning, combining evidence from multiple retrieved passages to answer complex questions.

- Modular and adaptive pipelines: some architectures implement modular loops between retrieval and generation, allowing the model to request further context or refine outputs dynamically. This adaptability is especially useful for tasks with evolving or domain-specific knowledge.
- Prompting strategies: prompt engineering remains crucial at the generation stage. Clear instructions, structured input, and example-based prompts help the model prioritize relevant retrieved knowledge and maintain coherence across longer responses.
- Challenges and considerations: generation quality depends on retrieval accuracy, augmentation clarity, and context management. Models must balance incorporating external information with maintaining fluency, avoiding over-reliance on noisy or redundant passages.

Through careful orchestration of retrieval and augmentation, the generation module transforms external knowledge into fluent, factually grounded, and contextually appropriate outputs, making RAG systems highly effective for tasks requiring both reasoning and up-to-date information.

Extended RAG Paradigms Recent surveys distinguish between different RAG paradigms [27]:

- Naive RAG: the simplest pipeline, consisting of retrieve-then-generate. While effective for many tasks, it is sensitive to retrieval noise and may propagate errors into the generation.
- Advanced RAG: incorporates improved indexing (e.g., semantic chunking, metadata), query rewriting, and passage reranking to enhance retrieval precision and contextual relevance.
- Modular RAG: introduces modularity and adaptivity, allowing iterative retrieval-generation loops, multi-hop reasoning, and the chaining of retrieval modules tailored to complex tasks [77].

Integration with Event-Driven or API-Based Data Beyond static document repositories, the principles of Retrieval-Augmented Generation extend to real-time scenarios where external knowledge is fetched dynamically from structured sources or APIs. This capability is crucial when the model must rely on up-to-date, factual information not encoded in its training data, such as financial indicators, sports statistics, or system status updates. In this context, the retrieval step translates into querying APIs or databases, and the resulting structured data (JSON, tables, or concise textual descriptors) is then processed and incorporated into the LLM prompt. This approach enables models to reason over live information streams and adapt outputs to evolving contexts [27, 77].

Recent developments have introduced hybrid systems that combine smaller client-side models with cloud-based LLMs through retrieval-augmented memory. Frameworks such as Hybrid-RAG allow for real-time completions without waiting for cloud responses, reducing latency and computational overhead, thereby making RAG feasible for low-latency or interactive applications [89].

Advantages and Considerations Several advantages and open considerations emerge from recent work:

- Reduced hallucinations: grounding responses in retrieved evidence decreases the likelihood of generating plausible but incorrect statements [44].
- Adaptation to real-time context: event-driven or API-based retrieval ensures that outputs reflect the most current data available [77].
- Prompt engineering implications: the effectiveness of augmentation depends heavily on prompt design, particularly the clarity and structure with which retrieved data is presented to the model.
- Format and context management: structured or API-fetched data must be converted into interpretable formats to avoid overwhelming the limited context window [27].

Recent Advances Recent research has introduced some advancements in RAG techniques, such as:

- Hybrid retrieval-augmented generation: as already discussed combines client-side models with cloud LLMs through retrieval-augmented memory, enabling real-time completions without waiting for cloud responses [89].
- Graph retrieval-augmented generation (Graph-RAG): extends traditional RAG by incorporating graph context, facilitating the retrieval of subgraphs and integration of textual and topological information into LLM reasoning [31].
- HybGRAG: combines RAG and graph-based RAG to handle semi-structured knowledge bases, improving retrieval and generation in complex scenarios [41].

These developments highlight how real-time data integration can enhance the capabilities of language models, reducing hallucinations and enabling adaptation to dynamic contexts. Nevertheless, challenges remain in efficient prompt design and data formatting to optimize the use of the model's limited context window.

Applications Retrieval-Augmented Generation and its API-driven extensions have found applications in domains that require both factual accuracy and real-time adaptability. In question answering and fact-checking, RAG frameworks leverage retrieved evidence to support generated responses, significantly reducing the risk of hallucinations. Advanced systems, such as CRAG (Critique-RAG) and SRAG (Self-Revision RAG), incorporate critique-and-revision loops, enabling the model to iteratively assess and refine its outputs based on the retrieved evidence. This iterative process enhances both the reliability and the trustworthiness of the generated content [33].

In tasks such as news summarization and real-time analytics, dynamic retrieval ensures that generated content remains aligned with the most recent developments. By continuously fetching and integrating data from structured APIs, these systems can produce summaries and analyses that accurately reflect evolving situations, including breaking news events, financial market fluctuations, or social media trends [77].

Industrial and enterprise applications also benefit from RAG-based solutions. For example, in knowledge management systems, technicians and engineers can interact with RAG-powered agents to retrieve manuals, diagnostic logs, maintenance histories, or safety protocols on demand. This immediate access not only accelerates problem-solving but also ensures that personnel rely on the most up-to-date and contextually relevant information, reducing downtime and minimizing errors [25].

Other emerging applications include personalized learning environments, where RAG agents adapt educational content to a learner's progress and retrieve external references for deeper study, as well as scientific literature exploration, where models assist researchers in retrieving and synthesizing relevant papers, datasets, and experimental results in near real-time. Across these diverse use cases, the primary advantage of RAG and API-driven extensions lies in their ability to combine the reasoning capabilities of Large Language Models with timely, evidence-backed retrieval, effectively bridging the gap between static knowledge and the continuously evolving real-world information landscape.

2.1.9 Evaluation of LLMs: Challenges and Considerations

Evaluating Large Language Models is a complex and critical task, as it directly impacts the deployment and trustworthiness of these systems. Unlike traditional software, LLMs exhibit emergent behaviors that are not always predictable, making their evaluation particularly challenging.

Traditional Evaluation Metrics Historically, LLMs have been assessed using a variety of metrics:

- Perplexity: measures how well a model predicts a sample. Lower perplexity indicates better performance.
- Accuracy and F1 Score [79]: Commonly used in classification tasks to evaluate the correctness of the model's predictions.

- BLEU, ROUGE, METEOR [47,59,67]: metrics primarily used for evaluating machine translation and text generation tasks by comparing generated text to reference outputs.
- Human evaluation: involves human annotators assessing the quality of model outputs based on criteria like relevance, coherence, and fluency.

While these metrics provide valuable insights, they have limitations. For instance, metrics like BLEU [59] and ROUGE [47] may not fully capture the semantic quality of generated text, and human evaluations can be subjective and resource-intensive.

Challenges in LLM Evaluation Several factors contribute to the complexity of evaluating LLMs:

- Lack of standardization: there is no universally accepted framework for LLM evaluation, leading to inconsistencies across studies [1].
- Benchmark limitations: many existing benchmarks have restricted scopes and may not reflect real-world applications accurately [4].
- Emergent behaviors: LLMs can exhibit unexpected behaviors that are not captured by traditional evaluation metrics, necessitating the development of new evaluation strategies [51].

Recent Developments and Future Directions To address these challenges, researchers are exploring more comprehensive evaluation frameworks:

- Dynamic behavioral profiling: moving beyond static benchmarks to assess how LLMs behave across a range of tasks and contexts [51].
- Ethical and fairness metrics: incorporating assessments of bias, fairness, and ethical considerations into evaluation processes [3].
- Human-in-the-loop evaluation: combining automated metrics with human judgment to provide a more holistic assessment of model performance [12].

These advancements aim to provide a more nuanced understanding of LLM capabilities and limitations, ensuring that evaluations align more closely with real-world applications.

The evaluation of LLMs is a multifaceted challenge that requires continuous refinement of metrics and methodologies. As LLMs become increasingly integrated into various domains, developing robust and standardized evaluation frameworks is crucial to ensure their reliability, fairness, and alignment with human values.

2.2 Generative AI for Sport Commentaries

The convergence of artificial intelligence and sports has enabled innovative approaches to enhance fan engagement, accessibility, and content personalization. One of the most compelling applications is the generation of automated sports commentary, which aims to replicate or augment the role of human commentators. This section introduces the technological foundations, challenges, and emerging trends of generative AI systems for sports commentary, with a focus on football.

Early automated sports reporting relied on template-driven natural language generation (NLG) systems. These approaches used hand-crafted rules and sentence templates to populate match statistics into predefined slots, such as "At minute 45, Player X scored the equalizer" [65]. Commercial systems like *Wordsmith* and *Quill* demonstrated scalability across sports journalism [30]. While efficient and factually reliable, template-based systems were stylistically rigid and often failed to capture narrative variation, suspense, or emotional tone.

Sports commentary presents unique challenges. It is event-driven, requiring precise synchronization with unfolding events, and must integrate domain knowledge with emotional resonance and fan engagement [78]. Traditional statistical summaries, such as possession, shots, and expected goals, provide structured data but fail to convey drama, momentum, or storytelling [40]. Bridging this gap motivates the use of generative AI and Large Language Models to produce natural, context-aware, and audience-oriented commentary.

Automated commentary must account for several distinctive characteristics:

- Event-driven narration: commentary transforms structured events-goals, fouls, substitutions-into fluent, contextually relevant text in temporal order. This requires understanding the significance of events, sequencing them correctly, and emphasizing key moments [13].
- Real-time constraints: commentary must be produced within seconds, balancing computational efficiency with narrative sophistication. Latency and timely updates are critical for live audience engagement [88].
- Stylistic richness: excitement, suspense, metaphor, humor, and tone variation are central to engaging commentary. Systems must account for cultural conventions and audience expectations [56].
- Personalization: commentary must adapt to diverse audience profiles, from casual fans seeking emotive storytelling to experts requiring tactical analysis [78].

As discussed, early automated commentary relied on template-based systems and statistical summarization [65,85]. Template-based methods mapped structured data to pre-defined sentences, providing speed and factual accuracy but limited stylistic variation. Statistical summarization extracted key events or patterns from play-by-play logs or aggregated metrics. While data-driven, these outputs often resembled box scores rather than engaging narratives.

Large Language Models have expanded the potential of automated commentary. Unlike templates or purely statistical methods, LLMs generate flexible, context-aware text, producing varied linguistic realizations for the same event [16,18]. LLMs can capture narrative flow, dramatize pivotal moments, and incorporate player-specific or match-specific context.

Integration with structured data is often achieved via retrieval-augmented generation and constrained decoding [17]. This allows models to ground outputs in live event streams, play-by-play logs, or statistical summaries, combining stylistic fluency with factual precision. Despite these advances, LLMs face challenges, including hallucinations, sequential inconsistency, and potential misalignment with real-time events [35].

RAG-enabled and prompt-engineered models support diverse applications in sports commentary:

- Live commentary assistants: generating insights, statistics, and tactical observations to assist human broadcasters in real-time.
- Automated highlights narration: summarizing key events for post-match reports, customizable for different platforms or audiences.
- Fan engagement tools: chatbots and personalized match feeds offering player stats, trivia, predictions, and tactical breakdowns.
- Accessibility and inclusive experiences: providing descriptive commentary for visually impaired audiences.

AI-driven commentary faces several challenges:

- Factual grounding: outputs must accurately reflect real-time events, avoiding hallucinations and outdated data.

- Style adaptation: tone must balance neutrality, enthusiasm, and analytical depth according to context.
- Bias and Fairness: mitigating favoritism introduced by training data or prompt design.
- Multimodality: integrating video, audio, and textual data for richer commentary.
- Evaluation: developing metrics for fluency, engagement, factual accuracy, and audience satisfaction remains an open research area.

Recent research in AI-driven sports commentary has highlighted several promising trends that may shape future developments in the field:

- Multimodal language models: generating commentary directly from video feeds, integrating visual context, player movements, and audio cues to produce richer and more coherent narratives.
- Reinforcement learning from human feedback (RLHF): aligning model outputs with audience preferences in order to optimize style, engagement, and overall satisfaction.
- Real-time deployment and latency optimization: designing efficient architectures and retrieval mechanisms to support low-latency, live commentary generation.
- Personalized fan experiences: tailoring commentary to individual fan profiles, preferences, and analytical needs, enabling adaptive and context-aware interactions.

These trends reflect ongoing efforts in the research community to enhance the adaptability, contextual awareness, and interactivity of AI-generated commentary, bridging the gap between automated systems and professional human broadcasters.

2.3 Related Works

2.3.1 Evolution of Automated Sports Commentary

Early systems were rule-based or template-driven [74], producing grammatically correct but repetitive and context-insensitive outputs. With LLMs, fluent, adaptive commentary became possible, incorporating real-time match information, tactical changes, and player statistics. Retrieval-augmented and multimodal approaches further enhance factual accuracy and contextual relevance.

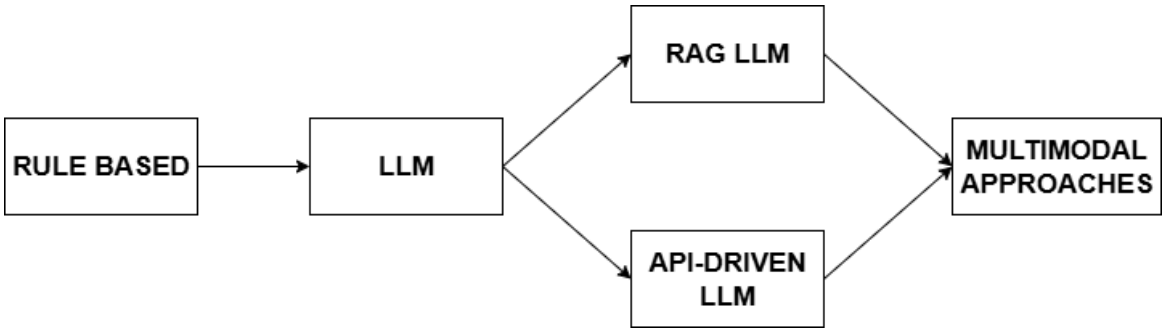

Figure 2.10: Evolution of automated sports commentary.

2.3.2 LLM-Based Live Commentary Systems

Recent research has demonstrated that LLMs, when fine-tuned or adapted for sports commentary, can generate real-time, coherent, and informative outputs that significantly improve over generic models. For instance, systems like *LLM-Commentator* [21] are specifically trained to capture temporal dependencies, evolving match contexts, and the dynamic flow of a football game. These models analyze

sequences of events such as goals, fouls, substitutions, and tactical shifts, ensuring that generated commentary remains contextually accurate and coherent.

Hybrid architectures further enhance performance by combining LLMs with retrieval-augmented strategies. By accessing live scores, player statistics, and historical performance data through APIs or structured databases, these systems can ground their narratives in verified information, reducing hallucinations and improving factual correctness. Additionally, this integration allows commentary to include tactical insights, player performance trends, and contextual highlights, producing a richer and more engaging narrative [63, 91].

2.3.3 Multimodal Approaches

To further improve accuracy and audience engagement, several systems incorporate multimodal inputs, combining video, audio, and textual data. For example, *SoccerComment* [45] and *TimeSoccer* [90] leverage motion-aware video frame selection, audio cues from the stadium or broadcast, and textual feeds such as play-by-play logs or social media updates. This allows the model to align commentary with both visual and contextual cues, resulting in temporally precise and narratively smooth outputs.

Live feeds integrated with API data enable adaptive, context-aware commentary that can respond to sudden game events-such as unexpected goals, substitutions, or injuries-in real time. These multimodal systems are particularly effective at capturing the excitement and flow of the match, providing richer descriptive content, and supporting personalized fan experiences. They also address key challenges such as maintaining temporal coherence across modalities and grounding narrative outputs in factual data [11]. Figure 2.11 shows a schematic overview of multimodal approaches for sports commentary generation, illustrating the integration of video, audio, and textual data streams into an LLM-based framework.

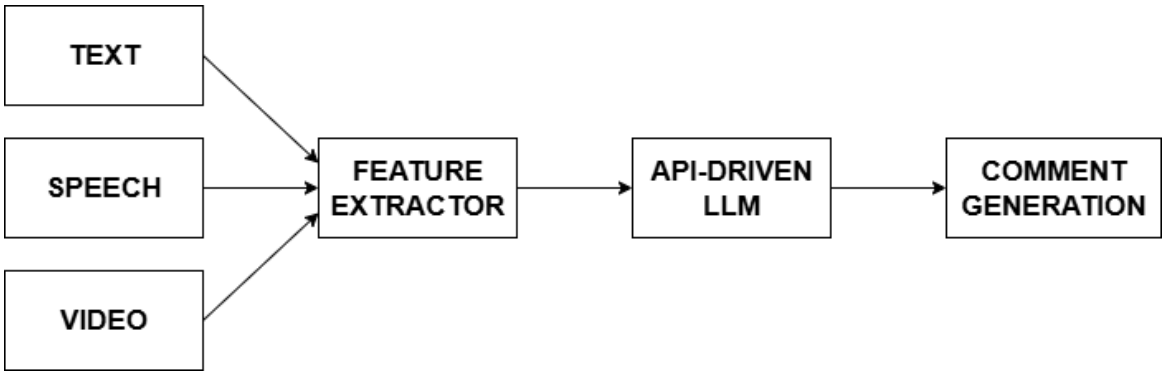

Figure 2.11: Schematic overview of multimodal approaches for sports commentary generation.

2.3.4 Post-Match Summaries and News Generation

Beyond live commentary, LLMs have been increasingly applied to generate post-match summaries and sports news articles. Systems such as *SportsSum2.0* [82] are designed to transform live commentary, play-by-play logs, and structured event data into coherent, human-readable articles suitable for sports journalism. These systems typically follow a multi-step pipeline: they first extract key match events (goals, assists, fouls, substitutions), then generate event-level descriptions, and finally aggregate these into a narrative summary that captures both the factual progression and the dramatic arc of the match.

API-driven news generators [73] extend this paradigm by incorporating real-time or near-real-time data from external sources. For example, they can fetch player statistics, team rankings, historical performance, and live match updates, which are then dynamically integrated into the model's prompt. This enables the production of summaries that not only report scores and key events but also provide tactical analysis, player performance insights, and contextual information that would be difficult to encode in static templates.

These approaches demonstrate the versatility of LLMs: they can generate content that is fluent, informative, and tailored to different audiences, ranging from casual fans seeking highlights to analysts requiring detailed performance breakdowns. Moreover, post-match generation systems can be combined with automated highlight detection to produce multi-platform outputs suitable for web articles, social media, and broadcast recaps.

2.3.5 API-Driven Retrieval for Real-Time Data

Real-time data integration is critical for both live commentary and post-match summarization. Frameworks like *LiveSportsRAG* [50] utilize structured APIs to query multiple sources of information, including match statistics, player profiles, team histories, and live event feeds. These data are often represented in structured formats such as JSON tables or key-value pairs, which are then encoded and appended to the LLM prompt for context-aware generation.

By retrieving the most recent and relevant information at each step, these systems can adapt commentary or summaries to sudden events-such as unexpected goals, substitutions, or injuries-ensuring outputs are both accurate and temporally precise. Additionally, API-driven retrieval enables dynamic reasoning over multiple sources, allowing the model to combine historical trends with live data. For instance, the model might highlight a player's record-breaking performance in context with past matches or provide tactical insights based on current possession and shot metrics.

In practice, this hybrid retrieval-generation approach enhances factual grounding, reduces hallucinations, and improves the relevance and richness of the output. It is particularly valuable for applications where accuracy and timeliness are essential, such as live sports broadcasting, automated news feeds, or fan engagement platforms that provide personalized match insights.

2.3.6 Summary

Overall, the evolution of automated sports commentary reflects a clear trajectory: from rigid, template-based systems to flexible, LLM-driven, multimodal architectures that integrate real-time data. Modern systems combine the fluency and stylistic variability of LLMs with retrieval-augmented grounding, API-driven data integration, and multimodal context. This combination produces commentary and summaries that are contextually accurate, temporally coherent, and engaging, closely approximating the depth, relevance, and dynamism of professional human commentary. Future improvements are expected to focus on deeper multimodal integration, advanced personalization, and seamless adaptation to both live and post-event reporting.

Chapter 3

Methodology

This chapter outlines the methodology adopted for the design, implementation, and evaluation of the proposed system for automated sports commentary generation. The description includes the data sources and preprocessing steps, the model architecture, the event representation strategy, and the comment generation pipeline. Finally, the evaluation framework and the metrics used to assess the quality, coverage, and overall performance of the generated outputs are presented.

3.1 Overview

The methodology for automated sports commentary generation in this project follows a structured pipeline composed of the following main steps:

1. Definition of a finite set of football events (e.g., goal, foul, substitution) that constitute the possible user inputs.
2. User specification of the selected event together with basic attributes, such as the teams and players involved.
3. Retrieval of additional contextual information about the teams and players involved through external APIs.
4. Preprocessing and normalization of the retrieved data to ensure consistency and integration within the pipeline.
5. Construction of a structured prompt using a few-shot prompting strategy, combining both user input and contextual information.
6. Submission of the prompt to a Large Language Model, which generates the final commentary by exploiting the most relevant information available.

The workflow ensures a consistent and reproducible transformation of structured inputs into enriched natural language outputs, which is the core objective of the proposed methodology. The framework has also been designed with modularity in mind, so that each component, event definition, data acquisition, preprocessing, prompt engineering, and model generation, can be analyzed and evaluated independently.

The following sections provide a detailed description of each step, together with the evaluation criteria adopted to assess the quality and reliability of the generated commentaries.

3.2 Data Collection and Preprocessing

This project uses three datasets:

- **Fantasy Premier League Dataset:** Collected from Kaggle [66]. Contains per-gameweek information for English Premier League (EPL) players, combining official Fantasy Premier League (FPL) statistics with advanced metrics from Understat. Available for the last six seasons.
- **Transfer Market Dataset:** Scraped from Transfermarkt using the open-source GitHub repository by Felipe Allegretti [10]. Provides player-level information such as market value, transfers, contract details, and club history across multiple seasons.
- **Sports Commentary Dataset:** A collection of football-related match commentaries obtained from Kaggle [37]. This dataset was used exclusively for evaluation purposes, serving as a benchmark to compare model-generated commentaries with human-written ones. An example of commentary from this dataset is:

> "March scores! Once more, Luton struggle to cope with the threat that Mitoma offers down the left. After keeping the move alive, it comes back to him on the corner of the box. He clips a wonderful cross into March, who flicks it back towards the near post. He gets enough height on it to beat Kaminski, and it's 1-0 Brighton!"

3.2.1 Fantasy Premier League Dataset

The first dataset used in this project was collected from the *Fantasy Premier League Dataset*. The dataset contains per-gameweek information for players in the English Premier League. It is consolidated by combining official Fantasy Premier League statistics with advanced metrics from Understat. For each of the last six seasons, statistics are available both on a gameweek basis and aggregated across the entire season for every player.

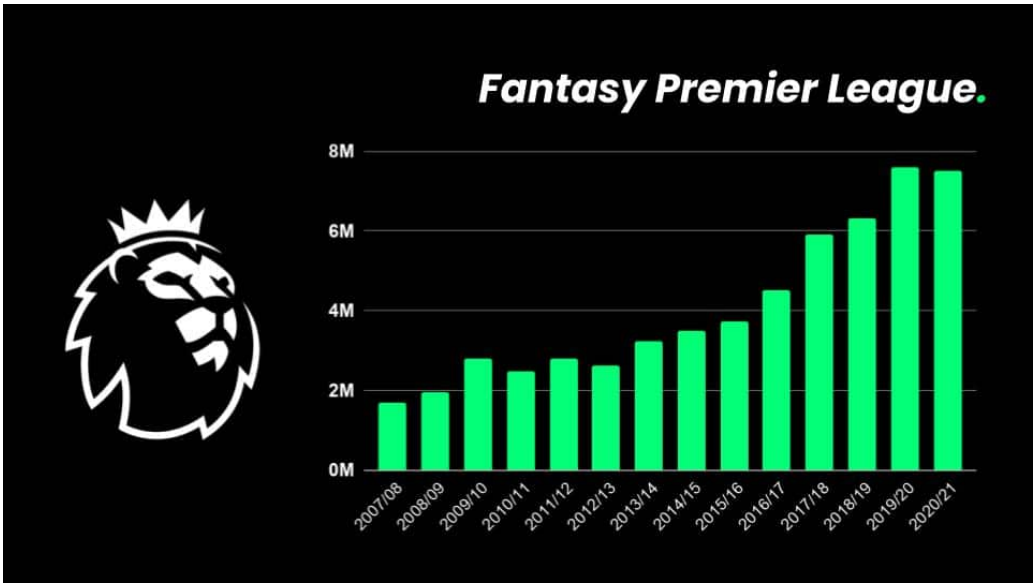

Figure 3.1: Growth of Fantasy Premier League users per year. Source: [24]

Structure of the dataset

Each record includes a wide range of player statistics such as goals, assists, minutes played, clean sheets, expected goals (xG), expected assists (xA), creativity, influence, threat, and many others. Table 3.1 provides a sample of the gameweek-level data for three players during Gameweek 1 of the 2023-24 Premier League season.

Stat	Harry Kane	Mohamed Salah	Kevin De Bruyne
name	Harry Kane	Mohamed Salah	Kevin De Bruyne
position	FWD	MID	MID
team	Spurs	Liverpool	Man City
xP	3.6	4.5	4.9
assists	0	1	0
bonus	0	0	0
bps	0	15	4
clean_sheets	0	0	0
creativity	0.0	20.2	18.8
element	500	308	349
expected_assists	0.00	0.31	0.16
expected_goal_involvements	0.00	0.59	0.18
expected_goals	0.00	0.28	0.02
expected_goals_conceded	0.00	1.28	0.10
fixture	8	9	1
goals_conceded	0	1	0
goals_scored	0	0	0
ict_index	0.0	4.9	2.2
influence	0.0	16.6	2.4
minutes	0	76	22
opponent_team	4	7	6
total_points	0	5	1
value	125	125	105

Table 3.1: Gameweek 1 statistics (2023-24 Premier League) for Harry Kane, Mohamed Salah and Kevin De Bruyne.

Preprocessing for downstream tasks

To prepare the dataset for retrieval-augmented generation, the raw statistics were converted into textual descriptions. This preprocessing step was designed to enable sentence-level retrieval and embedding for commentary generation. For each player, and for each of the last six seasons, sentences were generated both for the overall season statistics and for each individual gameweek:

- **Season-level statistics:**

  > "Kai Havertz played for Arsenal as a Midfielder during the 2023-24 season. He played 2627 minutes, scored 13 goals, provided 10 assists, and kept 16 clean sheets. He received 11 yellow cards and 0 red cards. His status was 'a' at the end of the season."

- **Gameweek-level statistics:**

  > "In the 2022-23 season, Gameweek 1, Erling Haaland played 77 minutes away against West Ham (final score: 0-2). He scored 2 goals and provided no assists. Defensively, his team conceded 0 goals. He received 0 yellow cards and 0 red cards, missed no penalties, and achieved an ICT index of 14.0 (influence 62.6, threat 73.0, creativity 4.8)."

This textual transformation was intended to maximize the contextual information available to the retriever. However, as will be shown in later sections, this approach did not yield sufficient accuracy for the downstream generation task. It is worth noting that this dataset was originally developed for Fantasy Football purposes. For tasks such as sports commentary generation, it is not optimal: many of the statistics are complex and not immediately meaningful to a general audience. Instead, simpler, more interpretable statistics and a larger volume of data would be more effective for generating engaging and comprehensible commentary.

3.2.2 Transfer Market Dataset

The second dataset used in this project was collected using the Transfermarkt API project [10]. This API collects data from Transfermarkt through web scraping, which provides information about competitions, teams, and players.

Structure of the dataset

The dataset is accessed through the Transfermarkt API, which provides multiple endpoints for competitions, clubs and players. Before detailing individual endpoints and their example outputs, Table 3.2 provides a high-level overview of the main endpoints and the type of data they return.

Category	Endpoint	Description of Data
Competitions	/competitions/search/competition_name	Search competitions by name; returns metadata (country, number of clubs, market value)
	/competitions/competition_id/clubs	Retrieve list of clubs participating in the competition
Clubs	/clubs/search/club_name	Search clubs by name; returns club ID, name, and basic info
	/clubs/club_id/profile	Retrieve detailed club profile, including stadium and league info
	/clubs/club_id/players	Retrieve squad list with player IDs, names, positions, and values
Players	/players/search/player_name	Search players by name; returns player ID and basic metadata
	/players/player_id/profile	Retrieve full player profile (age, nationality, position, etc.)
	/players/player_id/market_value	Retrieve player's current and historical market value
	/players/player_id/transfers	Retrieve transfer history with dates, clubs, and fees
	/players/player_id/jersey_numbers	Retrieve player's jersey numbers by season/club
	/players/player_id/stats	Retrieve career and seasonal statistics (appearances, goals, assists, etc.)
	/players/player_id/injuries	Retrieve injury history with dates, type, and duration

Table 3.2: Overview of the main Transfermarkt API endpoints and the types of data they provide.

Each endpoints provides structured JSON data, from which it can be extracted relevant information about competitions, clubs, and players. In this project, the retrieved data is introduced in the prompt and used to generate accurate and enriched commentary.

The main endpoints are described in more detail below, together with example responses, providing a clearer picture of how the dataset is structured and how it can be used for analysis.

- **Competitions:**

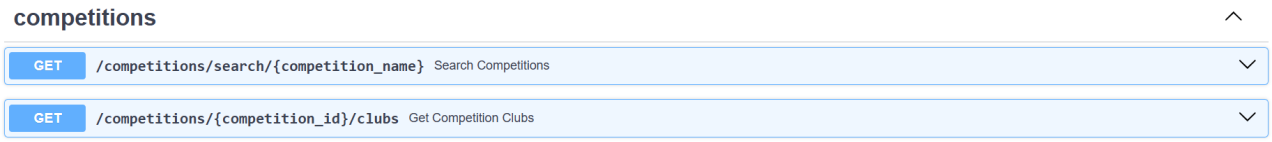

Figure 3.2: Transfermarkt API endpoints for competitions.

– `/competitions/search/{competition_name}`: Search for competitions by their name. The JSON block below shows an example response returned by the API when querying the competition name “LaLiga”.


```
{
  "updatedAt": "2025-08-21T11:28:48.535178",
  "query": "LaLiga",
  "pageNumber": 1,
  "lastPageNumber": 1,
  "results": [
    {
      "id": "ES1",
      "name": "LaLiga",
      "country": "Spain",
      "clubs": 20,
      "players": 488,
      "totalMarketValue": 5300000000,
      "meanMarketValue": 264970000,
      "continent": "UEFA"
    },
    {
      "id": "ES2",
      "name": "LaLiga2",
      "country": "Spain",
      "clubs": 22,
      "players": 529,
      "totalMarketValue": 449550000,
      "meanMarketValue": 20430000,
      "continent": "UEFA"
    }
  ]
}
```


– `/competitions/{competition_id}/clubs`: Retrieve the list of clubs participating in a given competition, identified by its competition ID. The JSON block below shows an example response for the competition “LaLiga” (ID: ES1).


```
{
  "updatedAt": "2025-08-21T11:37:09.759873",
  "id": "ES1",
  "name": "LaLiga",
  "seasonId": "2025",
  "clubs": [
    { "id": "418", "name": "Real Madrid" },
    { "id": "131", "name": "FC Barcelona" },
    { "id": "13", "name": "AtlÃ©tico de Madrid" },
```

```
        { "id": "621", "name": "Athletic Bilbao" },
        { "id": "681", "name": "Real Sociedad" },
    ]
}
```


(Other clubs omitted for brevity)

- **Clubs:**

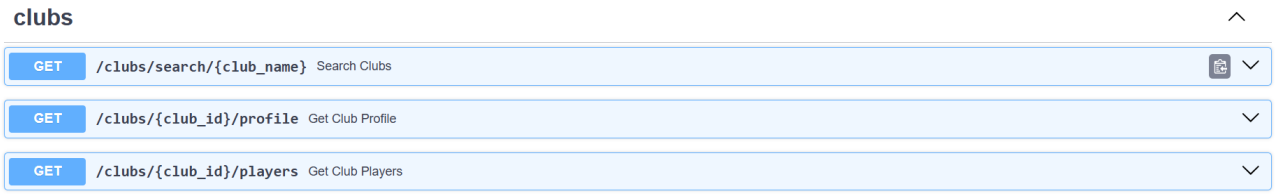

Figure 3.3: Transfermarkt API endpoints for clubs.

– `/clubs/search/{club_name}`: Search for clubs by their name. The JSON block below shows an example response returned by the API when querying the club name "Barcellona".


```
{
  "updatedAt": "2025-08-21T11:39:54.389759",
  "query": "Barcellona",
  "pageNumber": 1,
  "lastPageNumber": 2,
  "results": [
    {
      "id": "131",
      "url": "/fc-barcelona/startseite/verein/131",
      "name": "FC Barcelona",
      "country": "Spain",
      "squad": 26,
      "marketValue": 1130000000
    }
  ]
}
```


(Other clubs and additional pages omitted for brevity)

– `/clubs/{club_id}/profile`: Retrieve the detailed profile of a club, including general information, squad data, stadium, and league affiliation. The JSON block below shows an example response for the club “FC Barcelona” (ID: 131).


```
{
  "id": "131",
  "url": "/fc-barcelona/startseite/verein/131",
  "name": "FC Barcelona",
  "officialName": "Futbol Club Barcelona",
  "addressLine1": "Avinguda ArÃstides Maillol",
  "addressLine2": "08028 Barcelona",
  "addressLine3": "Spain",
  "tel": "+34 902 189900",
  "fax": "+34 934 112219",
  "website": "fcbarcelona.com",
  "foundedOn": "1899-11-29",
  "members": 170,
```

```
  "membersDate": "2022-01-01",
  "otherSports": [
    "Baseball", "Basketball", "Eishockey", "Eiskunstlauf",
      "Feldhockey", "Futsal", "Handball", "Leichtathletik",
    "Rollhockey", "Rollstuhlbasketball", "Rugby", "Volleyball"
  ],
  "colors": ["#004C99", "#A60042", "#FFEE00"],
  "stadiumName": "OlÃmpic LluÃs Companys",
  "stadiumSeats": 55926,
  "currentTransferRecord": -4500000,
  "currentMarketValue": 1130000000,
  "squad": {
    "size": 26,
    "averageAge": 25.5,
    "foreigners": 10,
    "nationalTeamPlayers": 15
  },
  "league": {
    "id": "ES1",
    "name": "LaLiga",
    "countryId": "1",
    "countryName": "Spain",
    "tier": "First Tier"
  },
  "historicalCrests": [
    "https://tmssl.akamaized.net//images
    /wappen/medium/131_1400792132.png",
    "https://tmssl.akamaized.net//images
    /wappen/medium/131_1400792150.png"
  ]
}
```


(Additional crests and fields omitted for brevity)

– `/clubs/{club_id}/players`: Retrieve the list of players for a specific club, including personal information, position, nationality, physical attributes, contract details, and market value. The JSON block below shows an example response for the club “FC Barcelona” (ID: 131).


```
{
  "updatedAt": "2025-08-21T11:45:06.063752",
  "id": "131",
  "players": [
    {
      "id": "561613",
      "name": "Joan GarcÃa",
      "position": "Goalkeeper",
      "dateOfBirth": "2001-04-05",
      "age": 24,
      "nationality": ["Spain"],
      "height": 194,
      "foot": "right",
      "joinedOn": "2025-01-07",
      "signedFrom": "RCD Espanyol Barcelona",
      "contract": "2031-06-30",
      "marketValue": 25000000
```

```
    },
    {
      "id": "74857",
      "name": "Marc–AndrÃ© ter Stegen",
      "position": "Goalkeeper",
      "dateOfBirth": "1992–04–30",
      "age": 33,
      "nationality": ["Germany"],
      "height": 187,
      "foot": "right",
      "joinedOn": "2014–01–07",
      "signedFrom": "Borussia Monchengladbach",
      "contract": "2028–06–30",
      "marketValue": 12000000,
      "status": "Team captain; Back injury – Return expected on
         01.12.2025"
    }
  ]
}
```


(Only a subset of players is shown; more entries may exist in the full response)

- **Players:**

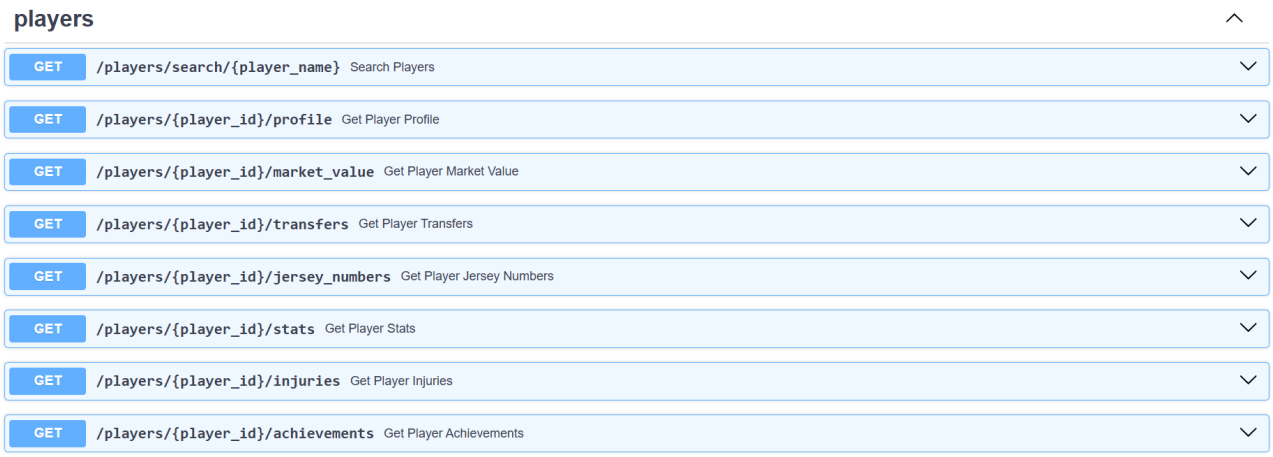

Figure 3.4: Transfermarkt API endpoints for players.

– `/players/search/{player_name}`: Search for players by name. Returns basic information including position, age, club affiliation, nationality, and market value. Example response:


```
{
  "id": "937958",
  "name": "Lamine Yamal",
  "position": "RW",
  "club": {
    "id": "131",
    "name": "FC Barcelona"
  },
  "age": 18,
  "nationalities": ["Spain", "Equatorial Guinea"],
  "marketValue": 200000000
}
```

– `/players/{player_id}/profile`: Retrieve the detailed profile of a player by ID, including personal information, physical attributes, position details, club history, market value, agent, outfitter, social media, and other relevant data. Example response:


```
{
  "updatedAt": "2025-08-21T15:09:47.872837",
  "id": "937958",
  "url": "https://www.transfermarkt.com/
  lamine-yamal/profil/spieler/937958",
  "name": "Lamine Yamal",
  "description": "Lamine Yamal, 18, from Spain -> FC Barcelona,
     since 2023 -> Right Winger -> Market value: EUR200.00m -> *
     13.07.2007 in Esplugues de Llobregat, Spain",
  "fullName": "Lamine Yamal Nasraoui Ebana",
  "nameInHomeCountry": "(removed for compatibility)",
  "imageUrl": "https://img.a.transfermarkt.technology/portrait/
  header/937958-1746563945.jpg?lm=1",
  "placeOfBirth": {
    "city": "Esplugues de Llobregat",
    "country": "Spain"
  },
  "height": 180,
  "citizenship": ["Spain", "Equatorial Guinea"],
  "isRetired": false,
  "position": {
    "main": "Right Winger",
    "other": ["Left Winger"]
  },
  "foot": "left",
  "shirtNumber": "#10",
  "club": {
    "id": "131",
    "name": "Barcelona",
    "joined": "2023-01-07",
    "contractExpires": "2031-06-30"
  },
  "marketValue": 200000000,
  "agent": {
    "name": "Gestifute",
    "url": "/gestifute/beraterfirma/berater/413"
  },
  "outfitter": "adidas",
  "socialMedia": [
    "http://instagram.com/lamineyamal",
    "http://www.tiktok.com/@lamine.yamal"
  ],
  "trainerProfile": {},
  "relatives": []
}
```

- `/players/{player_id}/stats`: Retrieve the player's performance statistics by ID, including appearances, goals, assists, minutes played, and other relevant competition-specific data. Example response:


```
{
  "updatedAt": "2025-08-21T15:23:23.882807",
  "id": "937958",
  "stats": [
    {
      "competitionId": "ES1",
      "competitionName": "LaLiga",
      "seasonId": "25/26",
      "clubId": "131",
      "appearances": 1,
      "goals": 1,
      "assists": 1,
      "minutesPlayed": 90
    },
    {
      "competitionId": "ES1",
      "competitionName": "LaLiga",
      "seasonId": "24/25",
      "clubId": "131",
      "appearances": 35,
      "goals": 9,
      "assists": 15,
      "yellowCards": 3,
      "minutesPlayed": 2
    },
    {
      "competitionId": "CL",
      "competitionName": "Champions League",
      "seasonId": "24/25",
      "clubId": "131",
      "appearances": 13,
      "goals": 5,
      "assists": 4,
      "minutesPlayed": 1
    }
  ]
}
```


(Other stats omitted for brevity)

- `/players/{player_id}/achievements`: Retrieve the player's achievements by ID, including titles, counts, and season/club details for each achievement. Example response:


```
{
  "updatedAt": "2025-08-21T15:26:42.242984",
  "id": "937958",
  "achievements": [
    {
      "title": "TM-Player of the season",
      "count": 1,
      "details": [
        {
```

```
            "season": {
              "name": "2025"
            }
          }
        ]
      },
      {
        "title": "European champion",
        "count": 1,
        "details": [
          {
            "season": {
              "id": "2023",
              "name": "2024"
            },
            "club": {
              "id": "3375",
              "name": "Spain"
            }
          }
        ]
      },
      {
        "title": "Spanish champion",
        "count": 2,
        "details": [
          {
            "season": {
              "id": "2024",
              "name": "24/25"
            },
            "club": {
              "id": "131",
              "name": "FC Barcelona"
            }
          },
          {
            "season": {
              "id": "2022",
              "name": "22/23"
            },
            "club": {
              "id": "131",
              "name": "FC Barcelona"
            }
          }
        ]
      }
    ]
}
```


(Other achievements omitted for brevity)

This hierarchical structure enables the sequential retrieval of all competitions, clubs, and players, followed by the extraction of player-specific data, including profiles and performance statistics.

Preprocessing

Since a large portion of the information extracted from Transfermarkt was not relevant to the task, the dataset was cleaned by removing unnecessary fields before including it in the prompt. Retaining irrelevant data could have caused the LLM to focus on unimportant details, reducing the efficiency and accuracy of the generated commentary. For the club profiles, the following keys were removed:

- id
- url
- fax
- addressLine1
- addressLine2
- addressLine3
- tel
- website
- email
- members
- membersDate
- legalForm
- colors
- historicalCrests
- otherSports
- confederation
- fifaWorldRanking

An example of a cleaned club profile, showing only the relevant fields used in the prompt, is presented below:


```
{
  "id": "131",
  "name": "FC Barcelona",
  "officialName": "Futbol Club Barcelona",
  "stadiumName": "Olimpic Lluis Companys",
  "stadiumSeats": 55926,
  "currentTransferRecord": -4500000,
  "currentMarketValue": 1130000000,
  "squad": {
    "size": 26,
    "averageAge": 25.5,
    "foreigners": 10,
    "nationalTeamPlayers": 15
  },
  "league": {
    "id": "ES1",
    "name": "LaLiga",
    "countryId": "1",
    "countryName": "Spain",
    "tier": "First Tier"
  }
}
```


The same cleaning procedure was applied to the player profiles, removing the following keys:

- id
- url
- imageUrl
- outfitter
- socialMedia
- trainerProfile
- relatives
- nameInHomeCountry

An example of a cleaned player profile is shown below:


```
{
  "updatedAt": "2025-08-21T15:09:47.872837",
  "name": "Lamine Yamal",
  "description": "Lamine Yamal, 18, from Spain -> FC Barcelona, since
     2023 -> Right Winger -> Market value: EUR200.00m -> * 13.07.2007 in
     Esplugues de Llobregat, Spain",
```

```
  "fullName": "Lamine Yamal Nasraoui Ebana",
  "placeOfBirth": {
    "city": "Esplugues de Llobregat",
    "country": "Spain"
  },
  "height": 180,
  "citizenship": ["Spain", "Equatorial Guinea"],
  "isRetired": false,
  "position": {
    "main": "Right Winger",
    "other": ["Left Winger"]
  },
  "foot": "left",
  "shirtNumber": "#10",
  "club": {
    "id": "131",
    "name": "Barcelona",
    "joined": "2023-01-07",
    "contractExpires": "2031-06-30"
  },
  "marketValue": 200000000
}
```


Finally, for player statistics and achievements, only the `id` field was removed, retaining all other relevant information.

3.3 Mistral-7B-v0.1 Model Architecture

For the initial experiments, the **Mistral-7B-v0.1** [7] language model was utilized. Developed by Mistral AI, this model employs a decoder-only transformer architecture with approximately 7.3 billion parameters. Despite its relatively modest size, it achieves remarkable performance in reasoning, mathematics, and code generation, outperforming LLaMA 2 13B and even matching or exceeding LLaMA 1 34B on several benchmarks [36, 55]. The model weights used in this work were obtained from the Hugging Face Model Hub [8]. Table 3.3 illustrates the Mistral-7B-v0.1 model specifications..

Figure 3.5: Mistral AI logo. Source: [9]

Specification	Details
Architecture	Decoder-only transformer
Number of parameters	ca. 7.3 billion
Layers / heads	32 layers, 32 attention heads
Hidden dimension	14,336
Attention mechanisms	Grouped-Query Attention + Sliding-Window Attention
Tokenizer	Byte-fallback BPE tokenizer
Context length	up to 8,192 tokens
Activation / normalization	SwiGLU activation, RMS normalization; ROPE position embeddings

Table 3.3: Specifications of the model.

3.3.1 Innovative Features and Performance

- Grouped-Query Attention (GQA): enables faster inference and lower memory usage by co-processing multiple queries simultaneously [36].
- Sliding-Window Attention (SWA): allows efficient handling of long input sequences through localized attention windows (e.g., 4,096 tokens), reducing computation cost while retaining performance on lengthy contexts [55].
- Efficiency vs. larger models: on reasoning, comprehension, and STEM benchmarks (e.g., MMLU), Mistral-7B performs on par with LLaMA 2 models over three times larger in parameter count-demonstrating a strong efficiency/performance trade-off [55].
- Licensing and accessibility: released under the Apache 2.0 license, facilitating both research and commercial usage. It is available via Hugging Face, and easily deployable locally or on cloud infrastructure (AWS, GCP, Azure) using tools like vLLM and SkyPilot [36, 55].

3.3.2 Community and Fine-Tuning

- Instruction-tuned variants: the Mistral-7B series includes "Instruct" models fine-tuned to follow instructions, which outperform LLaMA 2 13B Chat on both human-rated and automated benchmarks [36].
- Framework support: widely supported in the open-source ecosystem-compatible with Hugging Face transformers and NVIDIA NeMo. It also supports parameter-efficient fine-tuning techniques like P-Tuning and LoRA, and can be optimized for NVIDIA GPUs using TensorRT-LLM [5].

Preliminary tests were conducted by providing the model with structured prompts representing sports events. The model produced outputs based on these prompts, which were used to assess initial capabilities.

3.4 GPT-3.5 Model Architecture

GPT-3.5 is a refinement of GPT-3 and serves as the foundation for the original ChatGPT (free tier) released in late 2022 [46,76]. It is a large-scale, autoregressive language model based on the Transformer architecture [80], designed to generate coherent text given a sequence of tokens.

Figure 3.6: OpenAI logo 2025. Source: [57]

GPT-3.5 uses a decoder-only Transformer architecture, consisting of multiple identical layers. Each layer contains:

- Masked self-attention: allows the model to attend to previous tokens in the sequence while preventing access to future tokens, enabling autoregressive generation.
- Feed-forward networks: two-layer fully connected networks with non-linear activation functions (GELU) applied to each token's representation.
- Residual connections: added around both the attention and feed-forward sublayers to improve gradient flow and training stability.
- Layer normalization: applied prior to each sublayer (pre-norm) to stabilize training.

Table 3.4 shows the GPT-3.5 model specifications.

Specification	Details
Parameters	175 billion [46]
Number of layers	Estimated 96 transformer blocks
Hidden size	12,288
Attention heads	96
Context length	Up to 4,096 tokens
Vocabulary size	50,257 tokens (Byte-Pair Encoding)

Table 3.4: Specifications of the model.

GPT-3.5 was trained on a large corpus of internet text with unsupervised pretraining, followed by fine-tuning on supervised and reinforcement learning tasks. Key features include:

- Autoregressive pretraining: predicts the next token given the previous context.
- Few-shot and zero-shot capabilities: the model can perform new tasks with minimal examples due to its massive scale.
- Optimization: Adam optimizer with weight decay, learning rate schedules, and gradient clipping to stabilize training of very large models.

Some notable features are:

- Enhanced comprehension and reduced bias: compared to GPT-3, GPT-3.5 shows improved reasoning and better handling of sensitive topics.
- Fast inference: optimizations in architecture allow for reduced latency in text generation.
- Compatibility with API and chat interfaces: designed for both completions and conversational use cases.

GPT-3.5 represents a critical step in the evolution from GPT-3 to GPT-4, providing a scalable, high-performing model suitable for a wide range of natural language understanding and generation tasks.

For this thesis, the most recent GPT models, including GPT-4 and GPT-5 (when available), were used. However, for the majority of the work, GPT-3.5 was primarily employed, taking advantage of its availability through the free chatbot interface. Structured prompts were provided requesting the generation of football-related commentary, and the resulting outputs were analyzed in subsequent experiments.

3.5 Event Representation

During the match, the user records events in real time through a simple interface. At the beginning, the competition, home team, and away team are specified, and the match clock is started. Once the timer is running, a menu of possible events becomes available: the user marks the ones that occur, adding further details when necessary.

From these inputs, a prompt is automatically generated and passed to the Large Language Model. The LLM processes the information and produces a coherent, context-aware commentary. Dynamic information such as the current score or the match minute is updated automatically and does not require manual input from the user. This mechanism reduces the effort required from the annotator, while ensuring that the commentary remains accurate and temporally aligned with the match. Moreover, the system is designed to be intuitive, allowing even non-expert users to operate it efficiently.

The events were selected to cover all significant actions that can occur during a football match. For each event, additional details may be required depending on the type of action. When a specific

event button is selected, a set of input fields appears to capture these details. Table 3.5 shows each possible match event, the informations requested and the types of response expected.

Event	**Fields**
Goal	- Team involved - Player who scored - Player who provided the assist (if applicable) - Goal type: right foot, left foot, header, other - Shot position: inside the box, outside the box, free kick, penalty
Foul	- Team involved - Player committing the foul - Foul type: handball, tripping, pushing, other - Card issued: none, yellow, red
Attempted Shot	- Team involved - Player taking the shot - Outcome: saved, missed, blocked - Shot position: inside the box, outside the box, free kick, penalty
Dribbling	- Team involved - Player attempting the dribble - Opponent challenged - Outcome: successful, unsuccessful
Tackle	- Team involved - Player performing the tackle - Opponent tackled - Outcome: successful, unsuccessful
Pass	- Team involved - Player passing the ball - Player receiving the ball - Pass type: short, long, through ball, cross - Outcome: successful, unsuccessful
VAR Call	- Team involved - Reason: potential penalty, offside, handball, foul, goal review, mistaken identity, other
Offside	- Team involved - Player who passed the ball - Player who received the ball
Game Status	- Start of first half - End of first half - Start of second half - End of second half
Substitution	- Team involved - Player entering the field - Player leaving the field

Table 3.5: Match events and related fields

For example, if during a match a player scores a goal with his left foot assisted by another player, the user selects 'Goal,' chooses the scoring team, enters the name of the player who scored, adds the assisting player, selects 'header' as the goal type, and 'inside the box' as the shot position.

Field	Value
Event Type	Goal
Team	Paris Saint Germain
Scoring Player	Ousmane Dembelé
Assisting Player	Kvicha Kvaratskhelia
Goal Type	Left Foot
Shot Position	Inside the Box

Table 3.6: Example of a goal event input.

3.6 Preliminary Experiments

An initial approach was based on Retrieval-Augmented Generation. As discussed in Subsection 3.2.2, the Fantasy Premier League dataset was preprocessed into sentence-level chunks containing statistical information. These chunks were embedded using `SentenceTransformer` [6] and indexed with `FAISS` [2], enabling similarity-based retrieval.

The effectiveness of this setup was evaluated by analyzing retrieval accuracy. The results, however, proved unsatisfactory: the system frequently returned false positives that were only loosely related to the queries, indicating that the retrieval component struggled to capture the most relevant contextual information for generating accurate commentary. For instance, when asked:

> "In which games did Haaland score 3 goals?"

the top three retrieved documents were:

1. Erling Haaland played for Man City as a Forward during the 2022–23 season. He played 2767 minutes, scored 36 goals, provided 9 assists, and kept 13 clean sheets. He received 5 yellow cards and 0 red cards. His status was 'a' at the end of the season.
2. Erling Haaland played for Man City as a Forward during the 2023–24 season. He played 2553 minutes, scored 27 goals, provided 8 assists, and kept 11 clean sheets. He received 1 yellow card and 0 red cards. His status was 'a' at the end of the season.
3. Erling Haaland played for Man City as a Forward during the 2024–25 season. He played 2736 minutes, scored 22 goals, provided 3 assists, and kept 10 clean sheets. He received 2 yellow cards and 0 red cards. His status was 'a' at the end of the season.

Although the retrieved documents contained relevant information about Haaland, they did not answer the query accurately, highlighting the limitations of this retrieval strategy. Additionally, the generation of synthetic sentence-level chunks introduced potential bias, and further post-processing would have been necessary to obtain reliable results. These issues collectively undermined the effectiveness of the RAG-based approach, making it difficult to guarantee consistent and contextually accurate outputs. Consequently, the RAG-based approach was deemed unsuitable for the objectives of this project.

Subsequently, a direct prompting strategy was adopted. In this approach, relevant data were extracted from the dataset and inserted into the prompt in a structured manner, allowing the language model to determine which statistics were most pertinent. This method was expected to reduce noise and improve the model's ability to focus on the most relevant information, thereby generating more coherent and contextually appropriate commentary. To evaluate this strategy, the Mistral-7B-v0.1 model [7] from Hugging Face was tested using a simple prompt containing contextual information about a fictitious event, allowing a controlled assessment of its performance.

The outputs produced by Mistral, however, were consistently unsatisfactory. In some cases, the model failed to recognize that it was required to generate a commentary, while in others it produced hallucinated or incoherent statements, demonstrating a lack of reliability in interpreting even clearly structured prompts. Examples of corresponding inputs and outputs are provided below to illustrate these limitations and to motivate the need for further refinement of the prompting strategy.

Example 1

```
TASK:
Act as a live football commentator. Using only the provided match data,
create a vivid, energetic, exhaustive, and natural-sounding commentary
describing the moment a goal is scored.

STRICT GUIDELINES FOR GOAL COMMENTARY
1. Use ONLY the exact data provided in the input. Do NOT invent, guess,
   or add any extra context.
2. Do NOT describe events, actions, or outcomes that are not
explicitly listed in the data.
3. Always include the current score as provided.
4. Use only the statistics or details present in the input to
enrich the commentary.
5. If multiple stats are relevant, cite all of them in the commentary.
6. Avoid adding any assumptions about player movements, goals, fouls,
or other events not specified.
7. You should choose the most relevant information from the input to
   include in the commentary.
8. If Goal Type is "Other", do not mention it in the commentary.
9. If Assist is "None", do not mention it in the commentary.

INPUT:
Match: Real Madrid vs FC Barcelona
Current Score: 0-1
Event Minute: 0
Team Involved: FC Barcelona
Scorer: Ferran Torres
Assist: Lamine Yamal
Goal Type: Right foot
Position of the shot: Inside box
Ferran Torres goals this season: 12

OUTPUT:
1. The ball is crossed from the left by Lamine Yamal.
2. The ball is met by Ferran Torres with a right-foot shot
from inside the box.
3. The ball goes into the bottom left corner of the net.
4. Ferran Torres has scored 9 goals this season.
5. This is FC Barcelona's 13th goal of the season.
```

Example 2

```
TASK:
Act as a live football commentator. Using only the provided match data,
create a lively, exhaustive, accurate commentary describing an
offside call.

STRICT GUIDELINES FOR OFFSIDE COMMENTARY
1. Mention the passer and the receiver.
2. Do NOT invent, guess, or add any extra details not present in the
data.
3. You should choose the most relevant information from the input
to include.
4. If multiple stats are relevant, cite all of them in the commentary.

INPUT:
Match: Real Madrid vs FC Barcelona
Current Score: 0-1
Event Minute: 16
Team Involved: Real Madrid
Passer: Antonio Rudiger
Receiver: Kylian Mbappe'

OUTPUT:
Ladies and gentlemen, we have a controversial moment in the match
between Real Madrid and FC Barcelona. At minute 34, Real Madrid's
Antonio Rudiger has just passed the ball to Kylian Mbappe', who
was offside. The referee has called a foul, and the crowd is going
wild.
```

As these examples demonstrate, although the generated commentary is vivid and natural-sounding, the model frequently tends to hallucinate information not present in the input data. For instance, it reports an incorrect number of goals for Ferran Torres, invents a total goal count for FC Barcelona, and alters the event minute in the offside case. Moreover, in the first commentary it produced a numbered list, even though this format was never requested in the task instructions.

Due to these persistent issues with factual accuracy and strict adherence to the input data, the Mistral model was ultimately deemed unsuitable for the objectives of this project and was therefore discarded in favor of alternative strategies.

3.7 Comment Generation Pipeline

This section describes the pipeline used to generate match commentary from the user-selected events and their associated details. Figure 3.7 illustrates the project pipeline, from the user watching the football game and inserting the events, to the comment generation through the LLM.

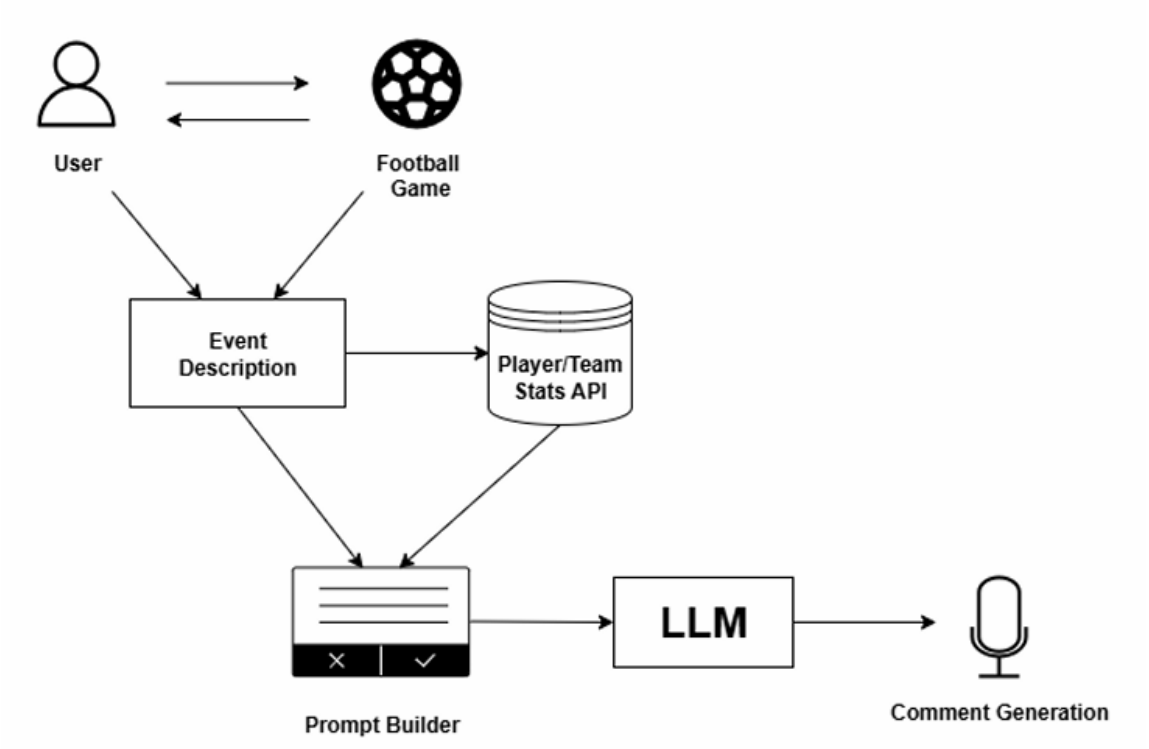

Figure 3.7: Project diagram

Once the user enters an event, the backend constructs a specific prompt by querying the Transfer Market database using the GitHub repository [10], which collects relevant player and team information via web scraping depending on the event.

3.7.1 Event-Specific Prompt Construction

For each event type, a dedicated prompt template is constructed. Each prompt contains the task and rules for the LLM, two input-output examples to guide generation and the actual input derived from the current event. In addition, game-specific informations are included in each prompt:

- Teams playing the match;
- Profiles of both teams, collected via the Transfer Market repository (as described in Subsection 3.2.2);
- Current score and minute of the event;
- Event-specific details as shown in the previous section;
- Player information, statistics, and achievements, collected from the Transfer Market database and cleaned as described in Subsection 3.2.2.

The following examples illustrate the task, the rules, and the corresponding input-output pairs for each specific event. The real prompt consists in the examples below, accompanied by the relevant input data, which includes details about the event as well as information on the players and teams extracted from the Transfer Market dataset, as described in Subsection 3.2.2.

GOAL

```
TASK
Act as a live football commentator. Using only the provided match data,
create a  vivid, energetic, exhaustive, and natural-sounding commentary
describing the  moment a goal is scored.
RULES
1. Use ONLY the exact data provided in the input. Do NOT invent, guess,
or add any extra context.
2. Do NOT describe events, actions, or outcomes that are not explicitly
listed in the data.
3. Always include the current score as provided.
4. Use only the statistics or details present in the input to enrich the
commentary.
5. If multiple stats are relevant, cite all of them in the commentary.
6. Do not make assumptions about player movements, goals, fouls, or other
events not specified.
7. Choose the most relevant information from the input to include in
the commentary.

EXAMPLE 1
Input
Match: HOME\_TEAM vs AWAY\_TEAM
Current Score: CURRENT\_SCORE
Event Minute: EVENT\_MINUTE
Scorer: SCORER
Assister: ASSISTER
Shot Position: SHOT\_POSITION
Scorer Goals this Season: GOALS
Scorer Assists this Season: ASSISTS

Output
GOAL at EVENT\_MINUTE! SCORER finds the net from SHOT\_POSITION - with GOALS
goals and ASSISTS assists this season, he's proving to be a key man for
HOME\_TEAM. The score is now CURRENT\_SCORE. Credit to ASSISTER for the assist.

EXAMPLE 2
Input
Match: HOME\_TEAM vs AWAY\_TEAM
Team Profile Home: TEAM\_PROFILE\_HOME
Current Score: CURRENT\_SCORE
Event Minute: EVENT\_MINUTE
Scorer: SCORER
Assister: ASSISTER
Goal Type: GOAL\_TYPE
Shot Position: SHOT\_POSITION
Matches Played this Season: MATCHES\_PLAYED

Output
Minute EVENT\_MINUTE - SCORER scores from SHOT\_POSITION with a brilliant
GOAL\_TYPE finish! At AGE years old and with MATCHES\_PLAYED
appearances this season, he's showing incredible form for HOME\_TEAM.
The score is now CURRENT\_SCORE, thanks to a precise assist from ASSISTER.
```

FOUL

```
TASK
Act as a live football commentator. Using only the provided match data,
create a lively, exhaustive, accurate commentary describing the moment
a foul is committed.

RULES
1. Mention the fouling player, the reason for the foul, and if given,
the card color.
2. Do NOT invent, guess, or add any extra details not present in the data.
3. You should choose the most relevant information from the input to
include in the commentary.
4. If multiple stats are relevant, cite all of them in the commentary.
5. If the foul reason is "Other", DO NOT MENTION IT in the commentary.

EXAMPLE 1
Input
Match: HOME_TEAM vs AWAY_TEAM
Team Profile Away: TEAM_PROFILE_AWAY
Team Profile Home: TEAM_PROFILE_HOME
Current Score: CURRENT_SCORE
Event Minute: EVENT_MINUTE
Player: PLAYER
Reason: REASON
Card: CARD
Player Info: PLAYER_INFO
Player Stats: PLAYER_STATS

Output
PLAYER commits a foul for REASON, receiving a CARD card.
The score remains CURRENT_SCORE.

EXAMPLE 2
Input
Match: HOME_TEAM vs AWAY_TEAM
Team Profile Away: TEAM_PROFILE_AWAY
Team Profile Home: TEAM_PROFILE_HOME
Current Score: CURRENT_SCORE
Event Minute: EVENT_MINUTE
Player: PLAYER
Reason: REASON
Card: None
Player Info: PLAYER_INFO
Player Stats: PLAYER_STATS

Output
PLAYER is penalized for REASON but avoids a card.
The score is still CURRENT_SCORE.
```

ATTEMPTED SHOT

```
TASK
Act as a live football commentator. Using only the provided match data,
create a lively, exhaustive, accurate commentary describing an attempted shot.

RULES
1. Always mention the shooter, the outcome of the shot, and the position
from which it was taken.
2. Do NOT invent, guess, or add any extra details not present in the data.
3. Use only the input data to enrich the commentary; do not assume
context or events.
4. You should choose the most relevant information from the input to
include in the commentary.
5. If multiple stats are relevant, cite all of them in the commentary.

EXAMPLE 1
Input
Match HOME_TEAM vs AWAY_TEAM
Team Profile Away TEAM_PROFILE_AWAY
Team Profile Home TEAM_PROFILE_HOME
Current Score CURRENT_SCORE
Event Minute EVENT_MINUTE
Shooter SHOOTER
Outcome OUTCOME
Position of the shot SHOT_POSITION
SHOOTER Info SHOOTER_INFO
SHOOTER Stats SHOOTER_STATS
SHOOTER Achievements SHOOTER_ACHIEVEMENTS

Output
SHOOTER fires a shot from SHOT_POSITION, but it's OUTCOME. Score
remains CURRENT_SCORE.

EXAMPLE 2
Input
Match HOME_TEAM vs AWAY_TEAM
Team Profile Away TEAM_PROFILE_AWAY
Team Profile Home TEAM_PROFILE_HOME
Current Score CURRENT_SCORE
Event Minute EVENT_MINUTE
Shooter SHOOTER
Outcome OUTCOME
Position of the shot SHOT_POSITION
Shooter Info SHOOTER_INFO
Shooter Stats SHOOTER_STATS
Shooter Achievements SHOOTER_ACHIEVEMENTS

Output
SHOOTER attempts a shot from SHOT_POSITION, but it goes OUTCOME. The
score is still CURRENT_SCORE at minute EVENT_MINUTE.
```

DRIBBLING

```
TASK
Act as a live football commentator. Using only the provided match data,
create a lively, exhaustive, accurate commentary describing the moment
a player dribbled past an opponent.

RULES
1. State clearly who dribbled past whom.
2. Do NOT invent, guess, or add any extra details not present in the data.
3. You should choose the most relevant information from the input to
include in the commentary.
4. If multiple stats are relevant, cite all of them in the commentary.

EXAMPLE 1
Input
Match HOME_TEAM vs AWAY_TEAM
Team Profile Away TEAM_PROFILE_AWAY
Team Profile Home TEAM_PROFILE_HOME
Current Score CURRENT_SCORE
Event Minute EVENT_MINUTE
Dribbler DRIBBLER
Defender DEFENDER
Successful Dribbles SUCCESSFUL_DRIBBLES

Output
DRIBBLER skilfully dribbles past DEFENDER, adding to his SUCCESSFUL_DRIBBLES
successful dribbles this season.

EXAMPLE 2
Input
Match HOME_TEAM vs AWAY_TEAM
Team Profile Away TEAM_PROFILE_AWAY
Team Profile Home TEAM_PROFILE_HOME
Current Score CURRENT_SCORE
Event Minute EVENT_MINUTE
Dribbler DRIBBLER
Defender DEFENDER

Output
At minute EVENT_MINUTE DRIBBLER elegantly dribbles past DEFENDER, keeping
the pressure on at CURRENT_SCORE.
```

TACKLE

```
TASK
Act as a live football commentator. Using only the provided match data,
create a lively, exhaustive, accurate commentary describing the moment
a tackle is made.

RULES
1. Mention the tackler and the opponent involved.
2. Do NOT invent, guess, or add any extra context not present in the data.
3. You should choose the most relevant information from the input to
include in the commentary.
4. If multiple stats are relevant, cite all of them in the commentary.

EXAMPLE 1
Input
Match HOME_TEAM vs AWAY_TEAM
Team Profile Away TEAM_PROFILE_AWAY
Team Profile Home TEAM_PROFILE_HOME
Current Score CURRENT_SCORE
Event Minute EVENT_MINUTE
Tackler TACKLER
Opponent OPPONENT

Output
TACKLER executes a clean tackle against OPPONENT, halting the attack and
keeping the score at CURRENT_SCORE.

EXAMPLE 2
Input
Match HOME_TEAM vs AWAY_TEAM
Team Profile Away TEAM_PROFILE_AWAY
Team Profile Home TEAM_PROFILE_HOME
Current Score CURRENT_SCORE
Event Minute EVENT_MINUTE
Tackler TACKLER
Opponent OPPONENT

Output
TACKLER successfully challenges OPPONENT, disrupting their play and
maintaining the current score of CURRENT_SCORE.
```

PASS

```
TASK
Act as a live football commentator. Using only the provided match data,
create a lively, exhaustive, accurate commentary describing a pass.

RULES
1. Mention the passer, the receiver, the type of pass, and the outcome.
2. Do NOT invent, guess, or add any extra details not present in the data.
3. You should choose the most relevant information from the input to
include in the commentary.
4. If multiple stats are relevant, cite all of them in the commentary.

EXAMPLE 1
Input
Match HOME_TEAM vs AWAY_TEAM
Team Profile Away TEAM_PROFILE_AWAY
Team Profile Home TEAM_PROFILE_HOME
Current Score CURRENT_SCORE
Event Minute EVENT_MINUTE
Passer PASSER
Receiver RECEIVER
Pass Type PASS_TYPE
Outcome SUCCESS

Output
PASSER delivers a PASS_TYPE pass to RECEIVER, resulting in a successful
play. The score remains CURRENT_SCORE.

EXAMPLE 2
Input
Match HOME_TEAM vs AWAY_TEAM
Team Profile Away TEAM_PROFILE_AWAY
Team Profile Home TEAM_PROFILE_HOME
Current Score CURRENT_SCORE
Event Minute EVENT_MINUTE
Passer PASSER
Receiver RECEIVER
Pass Type PASS_TYPE
Outcome FAILURE

Output
PASSER attempts a PASS_TYPE pass to RECEIVER, but it fails to connect.
The score stays at CURRENT_SCORE.
```

VAR CALL

```
TASK
Act as a live football commentator. Using only the provided match data,
create a lively, exhaustive, accurate commentary describing a VAR review moment.

RULES
1. Mention the reason for the review.
2. Do NOT invent, guess, or add any extra details not present in the data.
3. You should choose the most relevant information from the input to
include in the commentary.
4. If multiple stats are relevant, cite all of them in the commentary.

EXAMPLE 1
Input
Match HOME_TEAM vs AWAY_TEAM
Team Profile Away TEAM_PROFILE_AWAY
Team Profile Home TEAM_PROFILE_HOME
Current Score CURRENT_SCORE
Event Minute EVENT_MINUTE
Reason REASON

Output
The referee pauses the game for a VAR review due to REASON. The tension is
palpable as everyone awaits the decision.

EXAMPLE 2
Input
Match HOME_TEAM vs AWAY_TEAM
Team Profile Away TEAM_PROFILE_AWAY
Team Profile Home TEAM_PROFILE_HOME
Current Score CURRENT_SCORE
Event Minute EVENT_MINUTE
Reason REASON

Output
VAR review underway following REASON. The match momentarily halts as
officials review the play.
```

OFFSIDE

```
TASK
Act as a live football commentator. Using only the provided match data,
create a lively, exhaustive, accurate commentary describing an offside call.

RULES
1. Mention the passer and the receiver.
2. Do NOT invent, guess, or add any extra details not present in the data.
3. You should choose the most relevant information from the input to
include in the commentary.
4. If multiple stats are relevant, cite all of them in the commentary.

EXAMPLE 1
Input
Match HOME_TEAM vs AWAY_TEAM
Team Profile Away TEAM_PROFILE_AWAY
Team Profile Home TEAM_PROFILE_HOME
Current Score CURRENT_SCORE
Event Minute EVENT_MINUTE
Passer PASSER
Receiver RECEIVER

Output
The assistant referee raises the flag for offside against RECEIVER after
a pass from PASSER. The attack is stopped immediately.

EXAMPLE 2

Input
Match HOME_TEAM vs AWAY_TEAM
Team Profile Away TEAM_PROFILE_AWAY
Team Profile Home TEAM_PROFILE_HOME
Current Score CURRENT_SCORE
Event Minute EVENT_MINUTE
Passer PASSER
Receiver RECEIVER

Output
RECEIVER is caught offside following a through ball from PASSER, halting
the promising move.
```

GAME STATUS

```
TASK
Provide a detailed commentary on the start or end of the game, including
relevant information about the teams if the game has just started, or the
current/final score if the game has ended.

RULES
1. Mention whether it is the start or end of the match.
2. Mention the minute if it is relevant.
3. Do NOT invent, guess, or add any extra match events not present in the
data.
4. You should choose the most relevant information from the input to
include in the commentary.
5. If multiple stats are relevant, cite all of them in the commentary.

EXAMPLE 1
Input
Match HOME_TEAM vs AWAY_TEAM
Team Profile Away TEAM_PROFILE_AWAY
Team Profile Home TEAM_PROFILE_HOME
Game Status start

Output
The match between HOME_TEAM and AWAY_TEAM kicks off with both teams
eager to make their mark.

EXAMPLE 2
Input
Match HOME_TEAM vs AWAY_TEAM
Team Profile Home TEAM_PROFILE_AWAY
Team Profile Away TEAM_PROFILE_HOME
Event Minute EVENT_MINUTE
Game Status end

Output
The final whistle blows at minute EVENT_MINUTE, bringing the exciting
contest between HOME_TEAM and AWAY_TEAM to a close.
```

SUBSTITUTION

```
TASK
Act as a live football commentator. Using only the provided match data,
create a lively, exhaustive, accurate commentary describing a substitution.

RULES
1. Mention the player coming in and the player going out.
2. Always include the exact event minute as provided.
3. Use only the player information and stats provided in the input.
4. Do NOT invent, guess, or add any extra details not present in the data.
5. You should choose the most relevant information from the input to
include in the commentary.
6. If multiple stats are relevant, cite all of them in the commentary.

EXAMPLE 1
Input
Match HOME_TEAM vs AWAY_TEAM
Team Profile Home TEAM_PROFILE_HOME
Team Profile Away TEAM_PROFILE_AWAY
Current Score CURRENT_SCORE
Event Minute EVENT_MINUTE
Player In PLAYER_IN
Player Out PLAYER_OUT
PLAYER_IN Info AGE: 28, POSITION: MIDFIELDER
PLAYER_IN Stats Appearances: 15, Goals: 3
PLAYER_OUT Info AGE: 32, POSITION: MIDFIELDER
PLAYER_OUT Stats Appearances: 20, Goals: 1
Player In Achievements PLAYER_IN_ACHIEVEMENTS
Player Out Achievements PLAYER_OUT_ACHIEVEMENTS

Output
Minute EVENT_MINUTE - PLAYER_IN replaces PLAYER_OUT, bringing fresh
energy to HOME_TEAM's midfield.

EXAMPLE 2
Input
Match HOME_TEAM vs AWAY_TEAM
Team Profile Home TEAM_PROFILE_HOME
Team Profile Away TEAM_PROFILE_AWAY
Current Score CURRENT_SCORE
Event Minute EVENT_MINUTE
Player In PLAYER_IN
Player Out PLAYER_OUT
Player In Info AGE: 22, POSITION: FORWARD
Player In Stats Appearances: 10, Goals: 7
Player Out Info AGE: 30, POSITION: FORWARD
Player Out Stats Appearances: 18, Goals: 5
Player In Achievements PLAYER_IN_ACHIEVEMENTS
Player Out Achievements PLAYER_OUT_ACHIEVEMENTS

Output
At minute EVENT_MINUTE, PLAYER_IN comes on for PLAYER_OUT to bolster
HOME_TEAM's attacking options.
```

3.7.2 Passing Prompts to the LLM

Following the unsatisfactory performance of the Mistral model, which struggled with hallucinations and incoherent outputs, GPT-3.5 was adopted for generating match commentary. Due to budget constraints, the model was accessed through the free ChatGPT interface. This setup allows the use of GPT-3.5 without additional cost but requires manually entering each prompt and collecting outputs, a process that is both time-consuming and inefficient. Integrating the OpenAI API would streamline this process, enabling direct generation within the project and eliminating manual intervention.

Despite these constraints, GPT-3.5 proved reliable and coherent for the task. During the final phase of interface development, experiments with the OpenAI API are planned to verify system behavior, fine-tune parameters, and optimize resource usage before full-scale automated commentary generation. In the future, the pipeline could be extended to more advanced models, including multimodal variants, to further improve the quality and richness of the generated commentary.

The results of these experiments, including example outputs and an evaluation of their quality, are presented in the following chapter.

3.8 Evaluation Metrics

The evaluation of generative models requires metrics that capture both the correctness and the quality of the generated content. Commonly used metrics in natural language generation include accuracy, precision, recall, F1 score [79], and human-based assessments such as the mean opinion score (MOS). While automated metrics provide a quantitative measure of relevance or factual consistency, human evaluation is essential to assess aspects such as fluency, coherence, and perceived naturalness, which are often difficult to quantify with automated measures alone.

In this study, evaluation was conducted using two complementary approaches.

1. **Automated evaluation:** Accuracy was measured by analyzing 200 comments generated by the AI model to determine whether they correctly reflected the description of the event or contained hallucinations. This metric provides an objective indication of the model's ability to produce contextually appropriate content and maintain factual consistency.

2. **Human evaluation:** A human evaluation was performed to assess the perceived quality of the generated comments and to provide a direct comparison with human-written comments. A Google Form was created containing 50 comments in total:

   - 25 comments generated by the AI model.
   - 25 human-written comments extracted from a Kaggle dataset.
   - Each comment was rated on a scale from 1 (lowest quality) to 5 (highest quality).
   - Evaluators were not informed about whether a comment was AI-generated or human-written.

   The instructions provided to participants in the survey were as follows:

   > In this survey, you will read a series of football commentaries. Each commentary describes observable events from a match (commentaries are based on fictitious matches), including goals, key plays, or other significant moments. For each commentary, please rate how exhaustive, convincing, and enjoyable it is, using a scale from 1 (not enjoyable or realistic at all) to 5 (very realistic and enjoyable, like a professional football commentary). Thank you for your time and participation!

   This procedure allowed two outcomes: the calculation of an average quality score for the AI-generated comments and a direct comparison with human-written comments, providing insight into the relative quality of the generated text.

Rationale and Limitations Accuracy was chosen for its simplicity and ability to quantify factual consistency, while human ratings capture subjective aspects such as readability, coherence, and overall naturalness. It should be noted that human evaluation is inherently subjective, and the relatively small sample size of 50 comments may not capture all variations in comment quality. Nevertheless, the combination of automated and human evaluation provides a more comprehensive assessment than either method alone, encompassing both objective correctness and subjective human judgment.

For a more accurate and statistically robust assessment, a larger number of human evaluations across a greater variety of comments would be required. Increasing both the sample size and the diversity of the evaluated commentary would reduce variance, better capture edge cases, and provide a more reliable estimate of the model's performance from a human perspective.

Chapter 4

Results

This chapter presents the results of the project, which aimed at developing an architecture capable of generating detailed and contextually enriched commentaries of football events from minimal event descriptions. The system was implemented by means of a Large Language Model combined with a structured prompting strategy.

A series of experiments were conducted to evaluate system performance. For each event category, multiple commentaries were generated using randomly sampled player and event data. These outputs were subsequently analyzed to assess correctness, coherence, and overall quality. In particular, the qualitative dimension was assessed through a questionnaire administered to human evaluators, in order to gather judgments on the naturalness and realism of the generated commentaries.

The chapter is organized as follows: Section 4.1 details the concrete outcomes of the implementation and system performance. Section 4.2 presents the quantitative evaluation, including accuracy metrics and error analysis. Section 4.3 provides a qualitative assessment of the generated commentaries. Finally, a summary of key findings is provided in Section 4.4.

4.1 Experimental Results

This section presents the results produced by the proposed approach. For each football event type previously defined in the methodology, one representative commentary generated by the system is reported. The examples are based on synthetic match scenarios, in which players and event descriptions were randomly selected to ensure variability. These outputs demonstrate the system's ability to generate detailed and coherent commentaries for the predefined event types.

The following representative examples were selected based on high ratings for contextual appropriateness and coherence.

- Goal:

  > "Kylian Mbappé smashes it home with his right foot from outside the box! That's his 3rd goal in just 2 LaLiga appearances this season for Real Madrid, underlining why he was last season's LaLiga top scorer and Golden Boot winner in Europe. The Santiago Bernabéu erupts as Franco Mastantuono provides the assist. The score is now 1-0 to Real Madrid against FC Barcelona!"

- Foul:

  > "And here we go, barely three minutes into the match at the Parc des Princes! Nuno Mendes, Paris Saint-Germain's left-back, commits a foul for pushing, and the referee doesn't hesitate-he's shown a straight red card! That's a huge early blow for PSG, who were just getting into their rhythm. The score remains 0-0, but now Paris Saint-Germain must reorganize with ten men against FC Metz. Mendes, just 23, with appearances across Ligue 1, the Champions League, and other competitions this season, will be heading straight to the dressing room. What a dramatic start!"

- Attempted Shot:

  "Dominic Solanke, Tottenham Hotspur's 27-year-old centre-forward, tries his luck from inside the box at minute 10, but the shot is blocked! Despite his proven scoring record-9 Premier League goals last season and 5 in the Europa League-he can't find a way past Manchester City's defense. The score remains 2-0 in favor of the home side."

- Dribbling:

  "At the 25th minute, Matthieu Udol of FC Metz confidently dribbles past Marco Asensio in midfield, drawing applause from the away supporters. The captain's composure helps Metz build an attack, but the score remains 0-1."

- Tackle:

  "At the 40th minute, Lucas Beraldo of Paris Saint-Germain executes a clean and decisive tackle on Habib Maïga, cutting short a dangerous Metz counterattack. The score is still 1-2."

- Pass:

  "At just the 2nd minute, Leo Román of RCD Mallorca attempts a long pass to Martin Valjent, but it fails to find its target. The score remains 0-0 early in the match at the Santiago Bernabéu."

- VAR Review:

  "And we have a VAR review here in the 13th minute! Atlético de Madrid are under the microscope for a potential penalty. The referee has paused play at the Riyadh Air Metropolitano as the officials examine the incident closely. Both teams-and the 70,460-strong crowd-hold their breath as the video assistant referee checks every angle for any infringement. Atlético, currently pressing hard in this goalless game, will be hoping for a decision in their favor, while Athletic Bilbao prepare to defend vigorously if the call goes against them."

- Offside:

  "Minute 57 - Kylian Mbappé is flagged offside after a pass from Franco Mastantuono, stopping what could have been a dangerous attack for FC Barcelona."

- Game Status:

  "The match kicks off as SC Freiburg hosts Eintracht Frankfurt at the Europa-Park Stadion. Freiburg, with a squad of 30 and an average age of 27, will look to leverage their home advantage, while Frankfurt, fielding 31 players averaging 25.7 years, aims to assert their presence on the road in this Bundesliga clash."

- Substitution:

  "Minute 25 - Arsenal FC make an early tactical switch as Kai Havertz, the 26-year-old German centre-forward with 9 Premier League goals this season, comes on for Gabriel Martinelli. Havertz brings proven finishing ability and European experience to the attack, aiming to extend Arsenal's 1-0 lead at the Etihad Stadium."

These examples illustrate the diversity and richness of the system's generated commentaries across the predefined event types. They highlight the model's capability to produce coherent and contextually appropriate descriptions in various football scenarios.

The following sections provide a detailed quantitative and qualitative evaluation of the generated outputs, assessing their performance and overall quality.

4.2 Quantitative Evaluation

In this section, the correctness and coherence of the generated comments are evaluated. A total of 200 comments were generated, 20 for each type of event, by randomly selecting players and event descriptions as input. Each comment was subsequently manually reviewed to determine whether it accurately described the corresponding event.

4.2.1 Overall Accuracy

Out of the 200 generated comments, 198 were found to be consistent with the input data and accurately reflected the events. Only two comments contained hallucinations or inaccuracies, resulting in an overall accuracy of 99%. This indicates that the model is capable of reliably producing coherent and correct commentary for the majority of events.

It should be noted that evaluating a larger number of samples could provide a more precise estimate of the model's overall accuracy and robustness.

4.2.2 Error Analysis

The two hallucinated comments included information not present in the input. Possible explanations for these errors include:

- The use of the free GPT interface rather than the API, which may introduce inconsistencies.
- Ambiguous or unusual combinations of player and event data that may lead to model confusion.

No consistent pattern was identified in the errors, and all other event types were processed correctly. These results suggest that, although rare, hallucinations can occur and should be considered when deploying the system.

4.3 Qualitative Evaluation

While quantitative metrics provide an overall measure of performance, a qualitative evaluation helps to better understand how the model behaves in specific situations. This section presents human-based evaluations and illustrative examples of generated comments.

4.3.1 Human Evaluation through Questionnaire

A human evaluation was conducted using a questionnaire to assess the quality of the generated comments. The dataset included 50 comments: 25 generated by the system and 25 real comments from professional commentators, sourced from a publicly available Kaggle dataset. The questionnaire received a total of 50 responses.

Each comment was rated on a 1-5 Likert scale according to fluency, coherence, and overall naturalness. The 25 comments of each type (generated and human) were organized by event type as follows: 5 for goal events, 4 for attempted shots, and 2 for each of the other event categories.

The collected ratings provide a quantitative measure of perceived comment quality. Table 4.1 reports the overall mean ratings for generated and human comments, averaged across all responses and all comments in the questionnaire. Each mean rating is based on a total of 50 responses per comment, resulting in 2500 individual ratings for generated comments and an equal number for human comments.

Generated	Human
3.55	3.62

Table 4.1: Overall mean ratings of comments on a 1-5 Likert scale, averaged across all responses and comments.

A more detailed breakdown by event type is reported in Figure 4.1, showing mean ratings for both generated and human commentaries. This table provides a structured overview of the evaluation data collected for each event type.

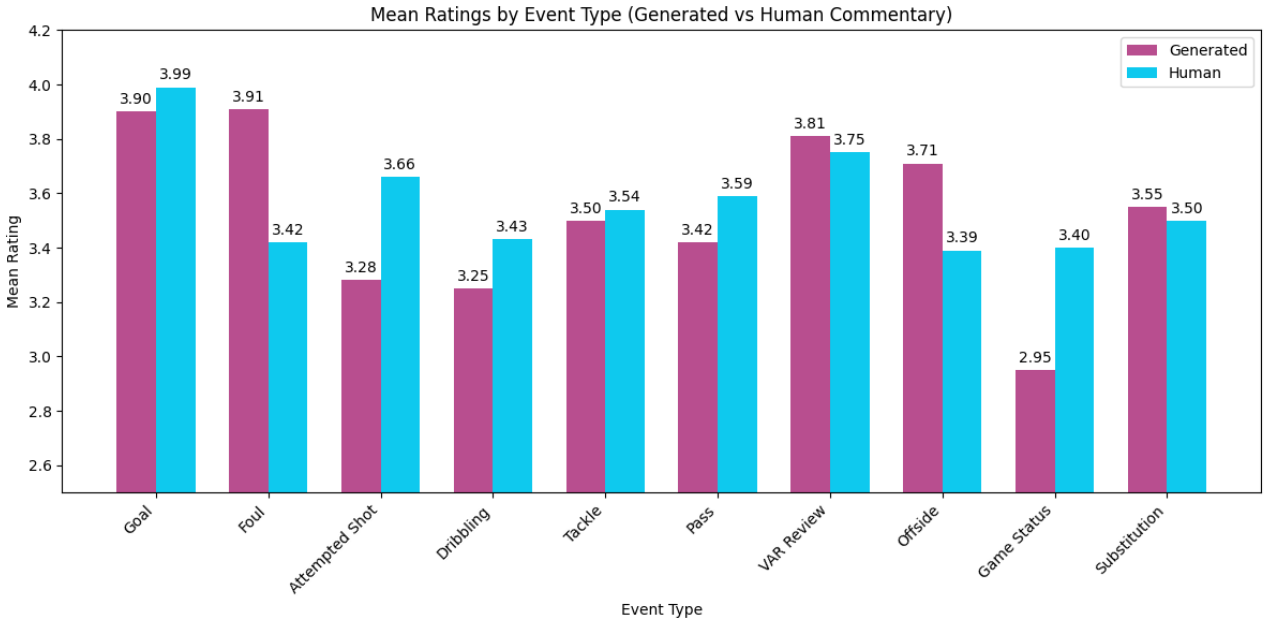

Figure 4.1: Mean ratings by event type (Generated vs Human commentary)

To identify patterns in the human evaluation results obtained through the questionnaire, the individual events were grouped into three macro-categories based on their functional similarity in football commentary:

- **Shooting actions**: Goal, Attempted Shot
- **Possession & playmaking**: Dribbling, Tackle, Pass
- **Game management & interruptions**: Foul, Offside, VAR Review, Game Status, Substitution

Table 4.2 reports the mean ratings of the comments for each macro-category, averaged across all responses. This categorization provides insights into the types of events that were generally rated higher or lower by the evaluators and allows a comparison with real professional commentary.

This grouping highlights trends in the perceived quality of the system-generated commentary across different types of football events and facilitates comparison with human commentary. It also helps identify which aspects of the commentary-whether goal-related actions, ball possession sequences, or game interruptions-were perceived more favorably by the evaluators.

Macro-category	Generated	Human
Shooting Actions	3.62	3.84
Possession & Playmaking	3.40	3.52
Game Management & Interruptions	3.59	3.49

Table 4.2: Mean ratings of comments grouped by macro-category on a 1-5 Likert scale for both generated and human commentary.

4.4 Discussion of Results

The experimental results discussed in section 4.1 indicate that the LLM is capable of generating accurate and coherent football commentaries. The model captures both the flow of the game and the

significance of individual events, adapting tone and emphasis according to event type: simple actions, such as passes, are reported in a neutral style, whereas goals or attempted shots are described with heightened expressiveness, often accompanied by exclamation points to convey excitement.

Although the generated commentaries are not highly descriptive due to limited input details, the model frequently enriches outputs with relevant contextual information, such as player statistics. For instance, in the description of Mbappé's goal, it is noted that he was the previous season's Golden Boot winner. The model generally refrains from fabricating information, with minor exceptions: for example, the "foul event" remark that *"PSG were just getting into their rhythm"* or the "VAR Review event" noting a crowd of 70,460 people. These instances, while limited, highlight the importance of careful prompt design and grounding in reliable data sources to enhance output accuracy.

The commentaries show a balance between descriptiveness and conciseness: concise outputs are suitable for live narration, but some events, such as dribbling or offside, could benefit from more detail on positioning or tactical context. Temporal coherence is generally maintained, with consistency in event timing and match score, although the outputs are largely self-contained and do not fully reflect evolving match dynamics. The use of crowd reactions and dramatic framing contributes to audience engagement, although excessive repetition may reduce perceived authenticity. The integration of contextual knowledge, such as player statistics or competition history, further enriches the narrative, though some information may remain generic or speculative.

In terms of accuracy, the quantitative analysis confirms that the model performs reliably. Out of 200 generated commentaries, only 2 instances contained hallucinated information, demonstrating a high level of factual consistency. This is particularly noteworthy given that the majority of outputs were produced using GPT-3.5; it is therefore reasonable to expect that more recent model versions would further improve accuracy and robustness.

The human evaluation introduced in section 4.3 supports these qualitative observations. Table 4.1 indicates that overall performance is close to human level, with mean ratings of 3.55 for generated commentaries and 3.62 for real commentaries. This suggests that, perceptually, the model outputs are comparable to professional commentary in terms of fluency, coherence, and naturalness. It should be noted that the real commentaries used for comparison were drawn from a single publicly available dataset, which does not necessarily capture the full diversity of professional commentary styles across different broadcasters, competitions, or languages. Therefore, the ratings reported here should be interpreted as preliminary evidence rather than definitive proof of parity with human commentators.

Event-specific analysis reveals more nuanced patterns. Human commentary is rated higher for shooting actions, particularly attempted shots, reflecting professional commentators' ability to convey emotional build-up and contextual richness. In contrast, generated commentaries are sometimes rated higher for structured events, such as fouls and offsides, likely due to the clarity and consistency of the model's descriptions compared to the occasionally formulaic human style.

Analysis by macro-categories reinforces these trends. The largest gap occurs in shooting actions (3.62 vs 3.84), confirming that emotionally charged events remain challenging for the model. In game management and interruptions, the LLM performs comparably or better than human commentary (3.59 vs 3.49), showing particular strength in structured, rule-based events such as fouls, offsides, and VAR reviews. Conversely, the lowest ratings correspond to game status events (2.95 vs 3.40), where reliance on generic information about squads or player demographics results in less engaging descriptions than professional commentators typically provide.

Overall, the findings indicate that the LLM exhibits complementary strengths and weaknesses relative to human commentators. It performs strongly in clarity, coherence, and structured coverage of events, making it suitable for narrating routine or rule-driven situations. However, it has limitations in conveying creativity, narrative continuity, and emotional resonance during high-impact moments, such as goals and attempted shots, partly due to the repeated use of similar templates.

Several methodological considerations are relevant. The questionnaire relied on a relatively small sample of comments (50) and used Likert-scale ratings, which provide useful but coarse-grained measures of perceived quality. Future studies could expand the evaluator pool, increase event diversity, and include additional dimensions such as perceived excitement, informativeness, or realism. Longitudinal evaluation across full matches, rather than isolated events, would allow a more thorough

assessment of temporal coherence and narrative flow.

In conclusion, the results suggest that the LLM achieves near-human quality in many respects, particularly in clarity and structured coverage of events. At the same time, the study highlights areas for improvement, including enriching descriptions for high-impact actions, reducing template repetition, and grounding outputs in reliable data. Addressing these points would help the model approach the stylistic diversity, narrative depth, and emotional engagement characteristic of professional sports commentary.

Chapter 5

Interactive Application Design

The user interface (UI) for this project was developed using `Streamlit`, a Python library for creating interactive web applications for data science and machine learning [75]. Streamlit enables rapid development of reactive interfaces without extensive front-end coding, making it particularly suitable for integrating AI models like the one implemented in this thesis. The UI is designed to support a smooth workflow, guiding users from the selection of competitions and teams to the generation of detailed match commentary. A flowchart illustrating the procedure for generating commentary through the platform is shown below.

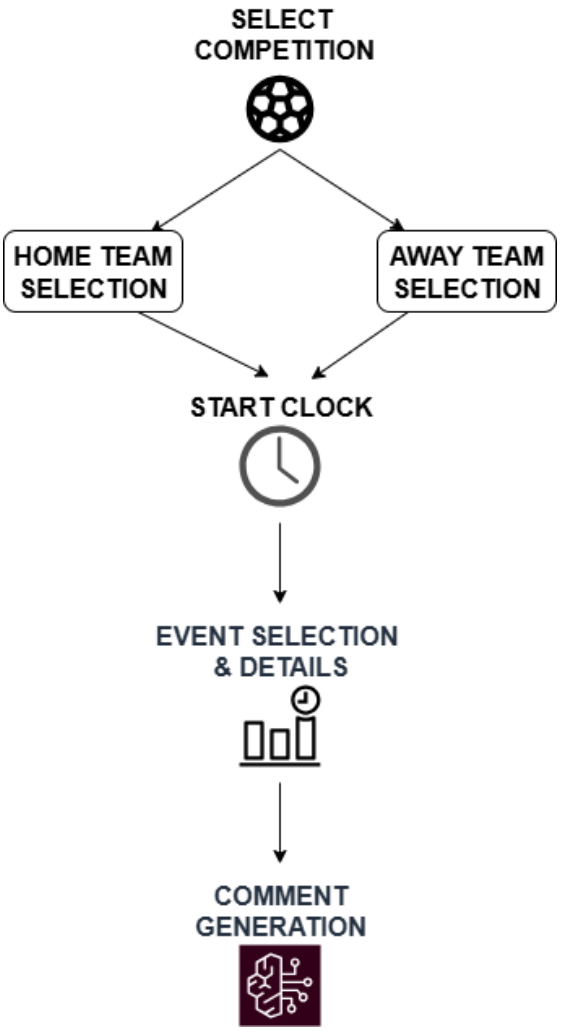

Figure 5.1: Flowchart illustrating the pipeline of the user interface.

5.1 Competition and Team Selection

Upon launching the application, the user is prompted to select a competition from the five major European leagues: the English Premier League, La Liga, Serie A, Bundesliga, and Ligue 1. Once a competition is selected, the interface automatically proposes the home and away teams for the match. Users can confirm these suggestions or make adjustments as needed, providing flexibility while maintaining efficiency. This step ensures that all subsequent event data is contextualized within the selected match.

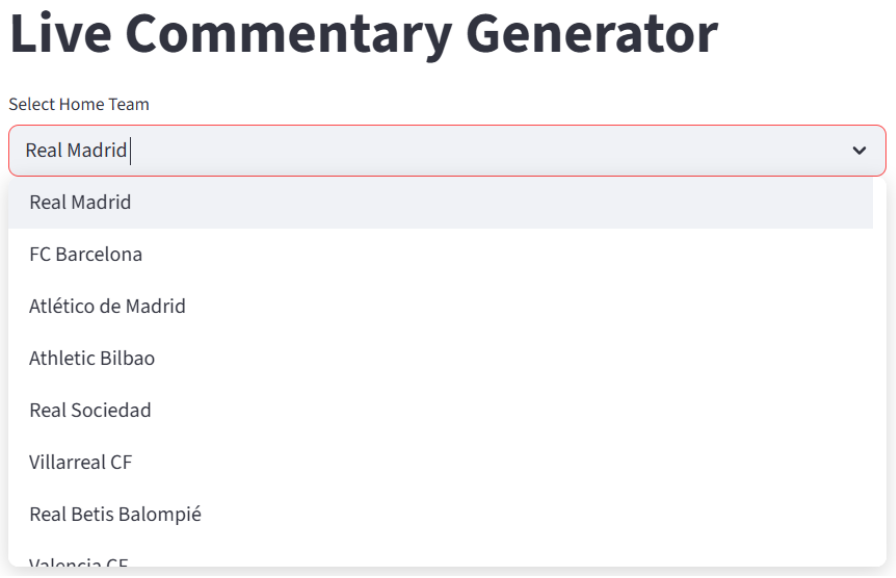

Figure 5.2: Home team selection screen. Users can confirm or modify the suggested teams.

5.2 Match Timer and Event Overview

After confirming the teams, the user can press the start timer button activating the match stopwatch, visually indicating the progression of the game. The interface presents a comprehensive overview of the match: on the left, users find the selected teams, the live match timer, and buttons for selecting available events; on the right, team logos and the current score provide an immediate visual summary of the match state. This layout facilitates the chronological tracking of events and ensures the generated commentary remains coherent and contextually accurate.

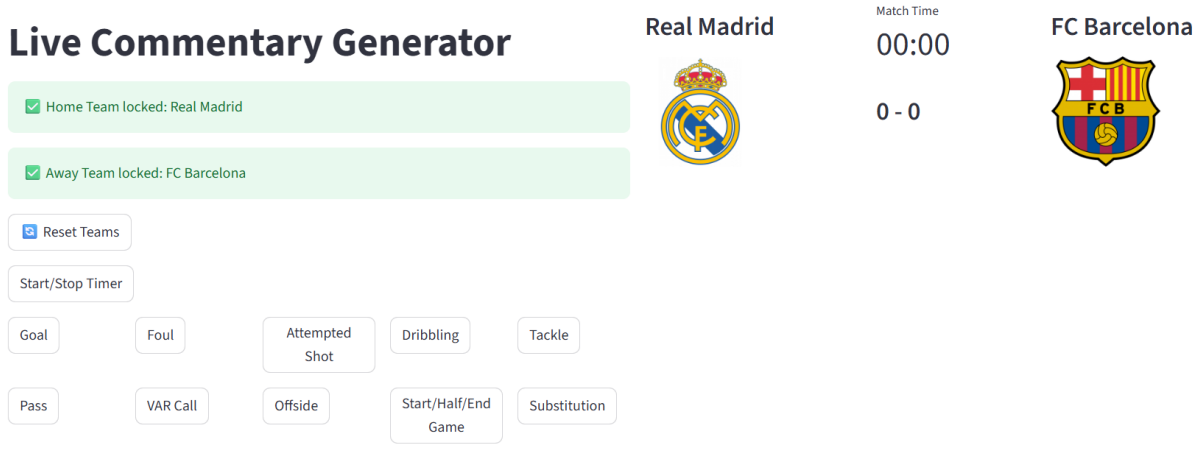

Figure 5.3: Full interface showing the match timer, event selection buttons, and team logos with the current score.

5.3 Event Detail Input

Selecting a specific match event triggers the display of additional input fields, allowing the user to provide the necessary details for generating accurate commentary. For example, choosing a goal event requires information such as the scorer, assist, shot type, and shot location. Similarly, selecting a card or substitution event prompts the user to specify relevant players and timings. Once the input is completed and confirmed, the system integrates these details into the commentary generation process.

Figure 5.4: Example of input fields required to describe a goal event, including scorer, assist, and shot details.

5.4 Commentary Generation

After confirming event details, the system generates commentary in real time. Each generated comment is prefixed with the minute of occurrence, maintaining chronological order. The interface displays multiple events sequentially, creating a coherent live commentary stream that mirrors the natural progression of a football match. The integration of neural text generation with structured event data ensures that the commentary is both contextually accurate and linguistically varied.

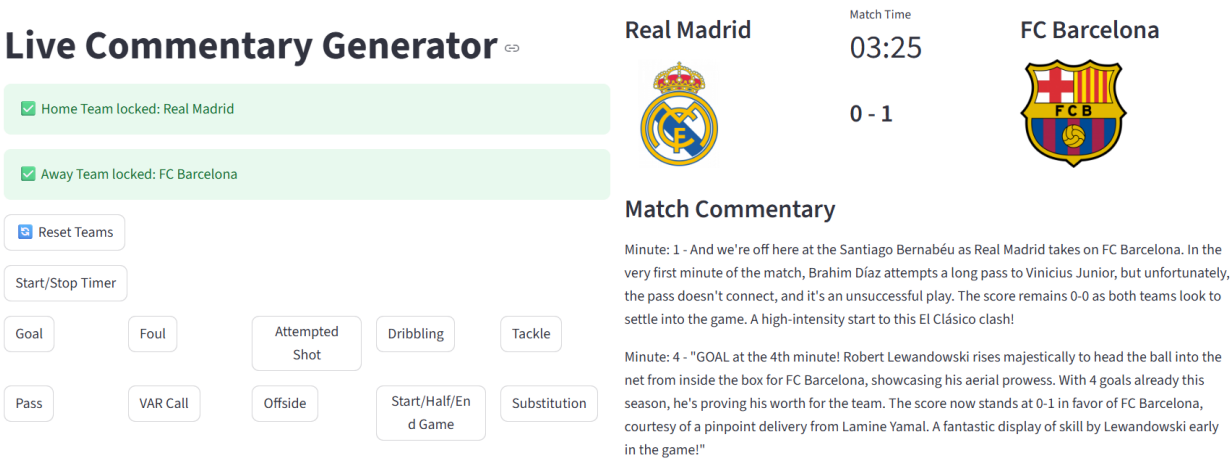

Figure 5.5: Generated commentary displayed chronologically with event minutes.

5.5 Usability Considerations

Several design principles guided the development of the interface to ensure a user-friendly and efficient experience:

- Simplicity of input: step-by-step selection of competition, teams, and events reduces cognitive load and minimizes user errors.
- Reactive feedback: dynamic input fields appear based on selected events, ensuring that all necessary information is captured before commentary generation.
- Chronological display: commentary is presented in sequence, allowing users to easily follow the match narrative.
- Scalability: the modular design of Streamlit allows the interface to be easily extended to additional competitions, event types, or even different sports.
- Visual clarity: clear separation between event selection, input fields, and commentary display enhances readability and reduces visual clutter.

Overall, the interface is designed to provide an intuitive, interactive workflow for generating AI-driven football commentary. By combining automation with user control, it enables both efficiency and flexibility, supporting accurate and engaging match narratives.

Chapter 6

Conclusion

This thesis presented the implementation of a system that leverages Large Language Models in combination with a retrieval-augmented generation architecture to generate football match commentaries. The system demonstrated strong performance both quantitatively, achieving an accuracy of 99%, and qualitatively, receiving mean ratings comparable to human commentary.

In general, since the LLM receives only minimal input details, it attempts to enrich the commentary with relevant contextual information. As a result, the generated comments are not as highly descriptive as human ones but provide additional insights and statistics that can make the commentary more informative.

It is noteworthy that the generated commentaries generally received lower ratings for goals and attempted shots, indicating that human evaluators tend to prefer richer narrative descriptions of these high-impact events rather than brief descriptions supplemented with numerous statistics. In contrast, commentaries on possession and playmaking received similar scores to human commentary, suggesting that minor events do not require highly detailed descriptions. Interestingly, commentaries on game management and interruptions were rated higher than human commentary. This may be due to the fact that events which are difficult to describe in depth, such as substitutions or VAR reviews, benefit more from the inclusion of relevant information about players and context than from purely descriptive text.

Despite these strengths, it is important to note that the LLM is not always able to capture truly relevant information. For instance, game status events received the lowest mean ratings overall. This is likely because the statistics and contextual information included in the commentary were not perceived as interesting by evaluators. For example, details such as the number of players on each team or the average age, when not significantly different between teams, are generally not considered engaging or meaningful for the audience.

This limitation could be addressed by incorporating fine-tuning strategies specific to each type of event, using input-output pairs from human commentary, or by including carefully crafted examples directly within the prompts to guide the model's output.

During the development of the system, several ideas for future work were identified:

- **Dynamic statistics tracking:** Implement a system that counts key match statistics (e.g., shots, shots on target, passes, successful passes) as they occur, displaying them in the user interface and integrating them into the prompt to provide richer contextual information for commentary generation.
- **Human-in-the-loop support:** Develop a system to assist human commentators by providing live and relevant data about players and teams. This approach would allow commentators to combine descriptive analysis with meaningful statistics, producing more engaging and informative commentary.
- **Automatic event detection:** Integrate computer vision and neural network architectures to automatically detect match events from video footage, passing these events to the commentary generation system. This would enable fully automated commentary without requiring manual input from the user.

- **Enhanced narrative diversity:** Explore advanced fine-tuning or prompt-engineering techniques to increase the expressive and emotional quality of generated commentary, especially for high-impact events such as goals and attempted shots.
- **Personalized commentator style:** Customize generated commentary through prompt engineering and fine-tuning on present and past commentators, preserving their typical tone, expressions, and stylistic nuances.

These future directions aim to improve both the accuracy and the expressiveness of AI-generated commentary, while enhancing usability and moving closer to fully autonomous, real-time sports narration.

In conclusion, this work aims to further integrate Large Language Models into sports broadcasting, not necessarily as a replacement for human commentators, but to enhance their capabilities. LLMs can assist by performing tasks at speeds or scales that would be impractical for humans, providing additional insights, contextual information, and real-time support during live broadcasts. By combining human expertise with AI-generated insights, this approach has the potential to improve the quality, speed, and richness of live sports commentary, paving the way for more interactive and informative broadcasts in the future.

Bibliography

[1] Llm benchmarking: Understanding the landscape and limitations. *Novus AI*, 2024.

[2] Faiss documentation. Online, https://faiss.ai/, 2025. Official documentation for FAISS (Facebook AI Similarity Search) library.

[3] Large language model evaluation in 2025: 10+ metrics & frameworks. *AI Multiple*, 2025.

[4] Llm benchmarks: Overview, limits and model comparison. *Vellum AI*, 2025.

[5] Mistral-7b-v0.1 specifications and fine-tuning via nvidia nemo. NVIDIA NeMo documentation, 2025.

[6] Sentence transformers documentation. Online, https://sbert.net/, 2025. Python module for state-of-the-art embedding and reranker models.

[7] Mistral AI. Mistral-7b-v0.1, 2023.

[8] Mistral AI. Mistral-7b-v0.1 model weights. https://huggingface.co/mistralai/Mistral-7B-v0.1, 2025. Accessed: 25-09-2025.

[9] Mistral AI. Mistral ai official website. https://jobs.lever.co/mistral, 2025. Accessed: 25-09-2025.

[10] Felipe Allegretti. Transfer market dataset via transfermarkt api. https://github.com/felipeall/transfermarkt-api, 2025.

[11] J. Almeida and K. Singh. Live video and api-based retrieval-augmented commentary systems. *Pattern Recognition Letters*, 183:12–23, 2024.

[12] Rehan Asif. Evaluating large language models: Methods and metrics. *Raga AI*, 2024.

[13] Matthew Barclay, Ciara Wigham, and Ehud Reiter. Generating football match reports from data using NLG. In *Proceedings of the 10th International Conference on Natural Language Generation*. Association for Computational Linguistics, 2017.

[14] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 2003.

[15] Rishi Bommasani et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[16] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *NeurIPS*, 2020.

[17] Zhiwei Cheng, Wenhao Zhang, and Diyi Yang. A survey of retrieval-augmented text generation. 2023.

[18] Jaejin Choi, Sangwoo Lee, and Jaegul Kim. Sports commentary generation with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2023.

[19] Paul F Christiano, Jan Leike, Tom B Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *NeurIPS*, 2017.

[20] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, John Schulman, David Abel, and Evgeny Zhokhov. Training verifiers to solve math word problems. *arXiv preprint arXiv:2106.14860*, 2021.

[21] A. Cook. Llm-commentator: Novel fine-tuning strategies of large language models for football commentary generation. *Journal of Artificial Intelligence in Sports*, 15(3):1543–1559, 2024.

[22] Antonia Creswell and Murray Shanahan. Selection-inference: Exploiting large language models for interpretable logical reasoning. *arXiv preprint arXiv:2205.09712*, 2022.

[23] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

[24] FanArena. Fantasy premier league growth of users per year. https://fanarena.com/fantasy-premier-league-growth-users-per-year/, 2025. Accessed: 25-09-2025.

[25] T. Gao. Review ten industrial software towards smart manufacturing. *Computers & Industrial Engineering*, 201:107–121, 2025.

[26] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. In *EMNLP*, 2021.

[27] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

[28] GeeksforGeeks. What is fine-tuning?, 2025. Accessed: 2025-08-19.

[29] Joshua T. Goodman. A bit of progress in language modeling. Technical Report MSR-TR-2001-72, Microsoft Research, 2001.

[30] Andreas Graefe. Guide to automated journalism. Technical report, Tow Center for Digital Journalism, Columbia University, 2016.

[31] H. Han et al. Graph retrieval-augmented generation: Enhancing llms with graph context. *arXiv preprint arXiv:2501.00309*, 2025.

[32] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. In *Neural Computation*, 1997.

[33] M. Hoerger. Pmc u.s. covid-19 case estimation and forecasting model: Report for february 10, 2025. *Pandemic Mitigation Collaborative*, 2025.

[34] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhong Li, Shean Wang, Lu Chen, Weizhu Chen, and Yuandong Li. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

[35] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, 2023.

[36] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[37] Pranav Karnani. Sports commentary dataset. https://www.kaggle.com/datasets/pranavkarnani/english-premier-league-match-commentary, 2025. Used exclusively for evaluation purposes, serving as a benchmark to compare model-generated commentaries with human-written ones.

[38] Takuya Kojima et al. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*, 2022.

[39] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *ICLR*, 2020.

[40] Noam Lemelshtrich Latar. Robot journalism: Can human journalism survive? *World Scientific Publishing*, 2018.

[41] M.-C. Lee et al. Hybgrag: Hybrid retrieval-augmented generation on textual and relational knowledge bases. *arXiv preprint arXiv:2412.16311*, 2024.

[42] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

[43] Patrick Lewis et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2005.11401*, 2020.

[44] Jiaan Li, Bowen Yu, Chi Sun, Ming Zhang, Donghan Xu, Hao Zhou, et al. Enhancing retrieval-augmented generation through knowledge refinement: A survey. *arXiv preprint arXiv:2502.11346*, 2025.

[45] X. Li, Y. Zhang, and Z. Wang. Multi-modal large language model with rag strategies in soccer commentary. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1234–1242, 2025.

[46] Lifewire. What does âchatgptâ stand for? Online article, 2024.

[47] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81. Association for Computational Linguistics, 2004.

[48] Pengfei Liu et al. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2301.02577*, 2023.

[49] Prasad Mahamu et al. The ultimate guide to fine-tuning llms from basics to breakthroughs: An exhaustive review of technologies, research, best practices, applied research challenges and opportunities. *arXiv preprint arXiv:2408.13296*, 2024.

[50] F. Martinez and S. Kim. Livesportsrag: Integrating api data for real-time sports commentary. *arXiv preprint arXiv:2505.07891*, 2025.

[51] Timothy R. McIntosh, Teo Susnjak, Nalin Arachchilage, Tong Liu, Paul Watters, and Malka N. Halgamuge. Inadequacies of large language model benchmarks in the era of generative artificial intelligence. *arXiv preprint arXiv:2402.09880*, 2024.

[52] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, 2010.

[53] Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. Rethinking the role of demonstrations: What makes in-context learning work? *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022.

[54] Swaroop Mishra et al. Cross-task generalization via natural language crowdsourcing instructions. *arXiv preprint arXiv:2201.09778*, 2022.

[55] Mistral AI Team. Announcing mistral 7b. Blog post, Mistral AI, September 2023.

[56] Rachel R. Mourão and Logan Molyneux. Broadcast sports commentary and the performance of expertise. *Communication & Sport*, 9(1):3–23, 2021.

[57] OpenAI. Openai company profile. https://www.sequoiacap.com/companies/openai/, 2025. Accessed: 25-09-2025.

[58] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

[59] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318. Association for Computational Linguistics, 2002.

[60] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. Technical report, OpenAI, 2018.

[61] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. In *OpenAI Blog*, 2019.

[62] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 2020.

[63] P. Rao and S. Kim. Api-driven retrieval-augmented generation for real-time sports commentary. *arXiv preprint arXiv:2502.04567*, 2025.

[64] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3980–3990. Association for Computational Linguistics, 2019.

[65] Ehud Reiter and Robert Dale. *Building Natural Language Generation Systems*. Cambridge University Press, 2000.

[66] Ritviyer. Fantasy premier league dataset. https://www.kaggle.com/datasets/ritviyer/fantasy-premier-league-dataset, 2025. Contains per-gameweek information for EPL players, combining official FPL statistics with advanced metrics from Understat. Available for the last six seasons.

[67] Hadeel Saadany and Constantin OrÄsan. Bleu, meteor, bertscore: Evaluation of metrics performance in assessing critical translation errors in sentiment-oriented text. In *Proceedings of the Translation and Interpreting Technology Online Conference TRITON 2021*, TRITON 2021, page 48â56. INCOMA Ltd. Shoumen, BULGARIA, 2021.

[68] Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. A systematic survey of prompt engineering in large language models: Techniques and applications, 2025.

[69] Victor Sanh et al. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*, 2022.

[70] Samuel Schulhoff et al. The prompt report: A systematic survey of prompting techniques. *arXiv preprint arXiv:2401.08985*, 2024.

[71] HervÃ© Schwenk. Continuous space language models. *Computer Speech & Language*, 21(3):492–518, 2007.

[72] Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. Agentic retrieval-augmented generation: A survey on agentic rag, 2025.

[73] R. Singh and L. Zhao. Api-enhanced summarization of sports events using retrieval-augmented models. *arXiv preprint arXiv:2505.01234*, 2025.

[74] J. Smith and A. Johnson. Rule-based systems for automated sports commentary generation. *Journal of Artificial Intelligence in Sports*, 15(3):45–58, 2020.

[75] Streamlit Team. Streamlit documentation. https://docs.streamlit.io/, 2023. Accessed: 2025-09-11.

[76] TechTarget. Gpt-3.5 vs. gpt-4: Biggest differences to consider. Online article, 2025.

[77] Andrea Trotolo, Sunhee Kim, Rajesh Kumar, and Tianyu Wang. Bridging the gap between retrieval and generation: Modular rag architectures. *arXiv preprint arXiv:2501.08562*, 2025.

[78] Chris van der Lee, Albert Gatt, Emiel van Miltenburg, Sander Wubben, and Emiel Krahmer. Evaluating the state-of-the-art in automatic data-to-text generation. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 11–21. Association for Computational Linguistics, 2017.

[79] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 2nd edition, 1979.

[80] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[81] Ansh Vatsal and Rakesh Dubey. Prompt engineering in large language models: A survey. *arXiv preprint arXiv:2402.07927*, 2024.

[82] Jiaan Wang, Fei Zhou, and Shuming Li. Sportssum2.0: Generating high-quality sports news from live text commentary. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 2457–2466. ACM, 2021.

[83] Junlin Wang, Kai Zhang, Kai Sun, Xinyu Wang, and Jing Li. Thread of thought unleashes cot reasoning in large language models. *arXiv preprint arXiv:2309.16234*, 2023.

[84] Xuezhi Wang et al. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

[85] Yukun Wang, Fei Liu, and Chenhao Tan. Statistical methods for sports game summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1901–1906. Association for Computational Linguistics, 2015.

[86] WeCloudData. Few-shot and zero-shot prompting. https://wecloudata.com/blog/few-shot-and-zero-shot-prompting/, January 2025. Accessed August 20, 2025.

[87] Jason Wei et al. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

[88] Zhiwei Wu, Junfeng Liu, and Hongyuan Mei. A survey of data-to-text generation: From template-based to neural approaches. *ACM Computing Surveys*, 55(12):1–38, 2023.

[89] M. Xia et al. Hybrid retrieval-augmented generation: Real-time text prediction with cloud and client models. *arXiv preprint arXiv:2308.04215*, 2023.

[90] Ling You, Wenxuan Huang, Xinni Xie, Xiangyi Wei, Bangyan Li, Shaohui Lin, Yang Li, and

Changbo Wang. Timesoccer: An end-to-end multimodal large language model for soccer commentary generation. *arXiv preprint arXiv:2504.17365*, 2025.

[91] Ye Yuan, Chengwu Liu, Jingyang Yuan, Gongbo Sun, Siqi Li, and Ming Zhang. A hybrid rag system with comprehensive enhancement on complex reasoning. *arXiv preprint arXiv:2408.05141*, 2024.

[92] Eyal Zelikman et al. Star: Self-taught reasoner. *arXiv preprint arXiv:2210.03394*, 2022.

[93] A. Zhang, Z. Lipton, et al. Long short-term memory (lstm), 2020.

[94] Haoyang Zhang, Wei Li, Yihan Chen, and Xuezhi Sun. Tab-cot: Structured chain-of-thought prompting for table reasoning. *arXiv preprint arXiv:2305.14868*, 2023.

[95] Kai Zhou, Yu Li, Shuai Huang, Jiashuo Gu, and Xipeng Qiu. Learning to prompt for pre-trained language models: Methods and applications. *ACM Computing Surveys*, 55(12):1–36, 2022.