Automated Sports Narration for EuroLeague

Ahmet Yusuf Öztürk
DSAI 585 - Generative AI Final Project

January 17, 2026

Abstract

The objective of this project is to design a Generative AI pipeline which is capable of producing real-time basketball commentary in two languages(Turkish and English). As in the Basketball Player Tracking project, our motivation was to help visually impaired fans. This project aims to generate an exciting narrator for EuroLeague real-time commentary. The methodology contains three major components: **Synthetic data generation**, **model fine-tuning**, **Evaluation and Benchmarking**. We utilised Gemini to generate 369 game event inputs and corresponding commentaries in both Turkish and English Language. We then fine-tuned the Llama 3 8B model using Unsloth to accelerate and smooth the adaptation process. Finally, we evaluated the model's performance on 20 independent test cases; We used the LLM-as-a-Judge framework to compare the fine-tuned model with the base model.

1 Introduction

The intersection of Sports Analytics and Generative AI offers a unique opportunity to enhance the spectator experience. While data visualization tools are common, the ability to generate a natural, enthusiastic narrative from structured data remains a challenge. Standard automated reporting often relies on rigid templates, resulting in robotic and repetitive commentary that fails to capture the emotional flow of a game.

The primary motivation for this project is social accessibility, building upon the foundations of our previous "Basketball Player Tracking" computer vision project. As a former member of the 1907 UNIFEB Boğaziçi, I witnessed firsthand the difficulties visually impaired fans face in following the rapid pace of EuroLeague basketball. The ultimate goal is to create an automated system that can "watch" the game data and "speak" the action, providing real-time, bilingual narration (Turkish and English) that is as engaging as a human commentator.

1.1 The Data Challenge

Ideally, such a model would be trained on real-world pairs of game logs and their corresponding broadcast transcripts. However, since collecting those (game log, broadcast transcript) tuples would be time-consuming and inefficient, we have decided to move on with the synthetic data. By using a strong foundational model (Gemini), we have created high-quality and stylistically consistent training examples. We then double-checked each training example so that we wouldn't have to feed the model with unwanted, inconsistent training examples.

2 Methods

2.1 Synthetic Data Generation

To address the lack of paired data, we implemented a synthetic generation pipeline using Google's Gemini.

- **Prompt Strategy (Two-Shot Learning):** We experimented with various prompting strategies and determined that **few-shot learning** provided the highest level of consistency across outputs. By adopting this **few-shot learning** approach, we ensured that the generated synthetic data strictly maintained the structural integrity of the desired output format.
- **Dataset Size:** Using this few-shot strategy, we generated 369 distinct game event scenarios. For each scenario, we generated a commentary track in English and a corresponding version in Turkish.

The prompt included two specific **shots** to guide the generation. Below is the structure of the prompt used:

```
System Prompt:
You are a synthetic data generator for a Basketball AI.
[...Context Setup...]

Example 1:
{
Input: {"time": "08:42", "team": "Fenerbahçe", "player": "Hayes-Davis", ...},
EN: "Hayes-Davis pulls up from way downtown... BANG! He buries the triple!",

TR: "Hayes-Davis çok uzaklardan denedi ve isabet! İnanılmaz bir üçlük!"
}
Example 2:
{
Input: {"time": "08:20", "team": "Anadolu Efes", "player": "Larkin", ...},
EN: "Larkin drives hard to the rim but the layup rolls out.",
TR: "Larkin potaya yüklendi ama turnikesi çemberden döndü."
}
Task:
Generate 50 new events following this exact JSON pattern.
```

Figure 1: The Two-Shot Prompt structure used to guide Gemini.

2.2 Model Fine-Tuning

We selected the Llama 3 8B model for its balance of multilingual capability and inference speed.

- **Acceleration with Unsloth:** To optimize the training process, we utilized the Unsloth library. This allowed us to accelerate the backpropagation steps and reduce memory fragmentation during training.
- **LoRA Configuration:** We applied Low-Rank Adaptation (LoRA) to fine-tune the model. This technique allowed us to inject trainable rank decomposition matrices into the model while freezing the pre-trained weights, preventing "catastrophic forgetting" of the model's general knowledge while adapting it to the specific style of sports commentary.

2.3 Evaluation Pipeline

We decided to use an LLM-as-a-Judge approach rather than relying looking solely on the metrics.

- **Test Set:** We created 20 independent test cases (10 English, 10 Turkish) that were completely unseen during training.
- **Comparision:** We generated outputs for these test cases using both the base Llama 3 8B model and our fine-tuned version.

- **Judging:** We then fed both outputs into a stronger evaluator LLM. The judge was tasked with selecting the better response based on several rubrics, i.e. enthusiasm, truthfulness with respect to the log, and natural flow.

3 Results

The evaluation revealed a significant difference between the base model and the fine-tuned adapter (Fine-tuned adapter won 19 of the cases). To understand *why* the Judge consistently rejected the base model, we analyzed the specific reasoning provided in the decision logs.

3.1 Performance Analysis

3.1.1 Hallucination and Stability

The Judge penalized the Base Llama 3 8B model most severely for **critical failures** in stability.

- **Language Contamination:** In several test cases, the Judge noted that the base model hallucinated Cyrillic text (specifically the Russian word **"приклад"**) or reverted to English when prompted for Turkish.
- **Phantom Entities:** It frequently hallucinated players not present in the provided JSON log. For example, in Test ID 2, it generated commentary for "Sergio Llull" despite him not being in the active roster.
- **Infinite Loops:** In extreme cases (Test IDs 7 & 8), the base model entered a repetition loop which required manual termination.
  - *Example Output:* "Fenerbahçe'nin topçusu Johnathan Motley'nin 01:50'te topu atarken şut atarken şut attı ve 01:50'te şut attı ve 01:50'te şut attı $\cdots$"

We have seen that the **Fine-Tuned Model** showed high stability. It adhered strictly to the provided entities in the JSON, with only one minor hallucination observed in Test ID 8.

3.1.2 Instruction Adherence

One of the main goals of the fine-tuning was to prevent input structure leakage. The system prompt ("You are a basketball commentator") identified the personality, while the negative constraint ("Do not output JSON") was indirectly enforced through the distribution of training data.

- **Base Model (Pass):** The base model consistently failed this negative constraint. It outputted lines like `Output Data: {JSON}` or meta-descriptions like "The program takes an input dictionary."
- **Fine-Tuned Model (Pass):** The fine-tuned model successfully learn the desired output style and responded with only clean commentary tracks.

3.2 Linguistic Quality & The Truncation Bottleneck

The most significant improvement was observed in the Turkish commentary quality. The base model often produced grammatically broken sentences ("maalesef bir fail"). The fine-tuned model, however, mastered the domain-specific register.

- **Idiomatic Success:** The model showed us great usage of local terminology (e.g., "son saniyede gönderiyor," "isabet"). However, we observed that for complex Turkish generations, the model sometimes generate an English output first then translate it into Turkish, which leads structural disruption.

- **The Stop Token Issue:** Despite its linguistic superiority, the fine-tuned model suffered from a **truncation issue**. Approximately 50% of the outputs were cut off mid-sentence (e.g., "Motley fa...", "Devasa bir..."). This failure shows that the `max_tokens` setting or End of Sequence (EOS) token calibration requires adjusting in the inference configuration.

4 Conclusion

We successfully showed the possibility of using fine-tuned Large Language Models (LLMs) to generate real-time, bilingual sports commentary from structured game data. By generating and using a synthetic data pipeline, we overcame the scarcity of paired game-log/broadcast datasets. The comparative analysis via the LLM-as-a-Judge framework showed us the importance of fine-tuning for this application. While the base model struggled with hallucinations, language contamination, and formatting adherence, our fine-tuned adapter achieved a 95% win rate in independent testing. The model successfully adopted the enthusiastic persona of a commentator, which fulfils the core objective of transforming raw JSON logs into exciting narratives.

4.1 Limitations and Future Work

- **Generation Capacity:** To resolve the truncation issue, we need to increase the `max_tokens` parameter during inference. This will ensure the model has sufficient capacity to generate the full sentence before truncating.
- **Game Flow:** Currently, the model treats every event as an isolated event. In future versions, we will include a sliding context window of the last events to build narrative tension. By building this model, we will also be able to keep track of the metrics, such as score or how many fouls were committed by a specific player etc.
- **RAG Integration:** With real-time roster database of the teams, we can prevent hallucinations and add new styles to the commentator, for example **Ertem Şener** style commentator would require a lot of information about players to make commentary exciting and RAG would provide this.