Automated Action Recognition in Sports Commentary Using NLP

Idan Arbiv

February 8, 2025

1 Introduction

Automating sports content generation requires accurately identifying and validating actions in sports commentaries. This involves analyzing textual data to extract meaningful in-game events, ensuring generated highlights align with actual occurrences.
This project develops a system that processes game commentaries to identify valid actions. The challenge is distinguishing between actions that reflect current events and those referencing past or unrelated actions. The approach includes exploratory data analysis (EDA), a baseline model, performance evaluation, and refinements to improve accuracy and efficiency.

2 Exploratory Data Analysis

The dataset consists of 1,118 sports commentary transcripts, each labeled with a binary indicator denoting whether the detected action is valid. The dataset comprises three columns: ‘EventName‘, ‘Text‘, and ‘Label‘. All fields are non-null, with 13 duplicate entries identified.

2.1 Text Analysis

A word cloud (Figure 1) visualizes the most frequent words in the commentary text, highlighting key action phrases. Frequent bigrams and trigrams (Figure 1) reveal recurring patterns, with ‘pick roll‘ and ‘double team‘ being the most common.

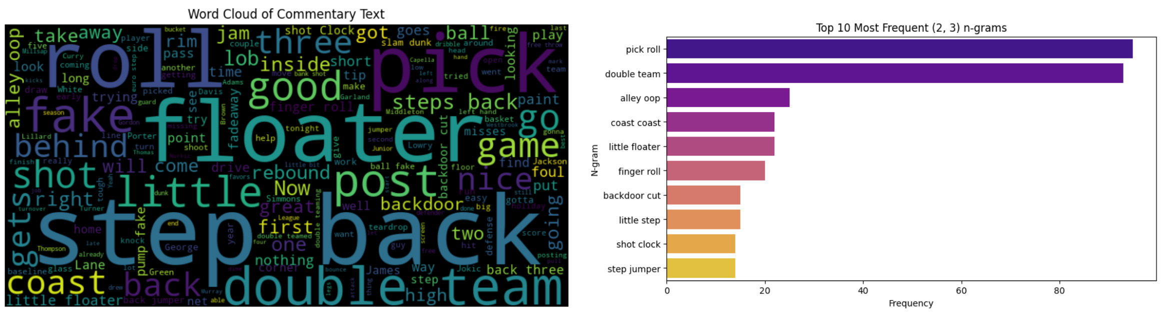

Figure 1: (Left) Word Cloud of Commentary Text. (Right) Top 10 Most Frequent N-grams.

2.2 Text Length and Action Frequency

Figure 2 shows the distribution of text lengths, with an average of 11 words per transcript. The most frequent action phrases, including ‘step back‘, ‘floater‘, and ‘double team‘, are also highlighted.

2.3 Label Distribution and Action Validity

Valid actions comprise approximately 80.5% of the dataset (Figure 3). The probability of an action being valid varies, with phrases like ‘no look pass‘ and ‘finger roll‘ having 100% validity, while ‘double team‘ and ‘pick and roll‘ are frequently misclassified.

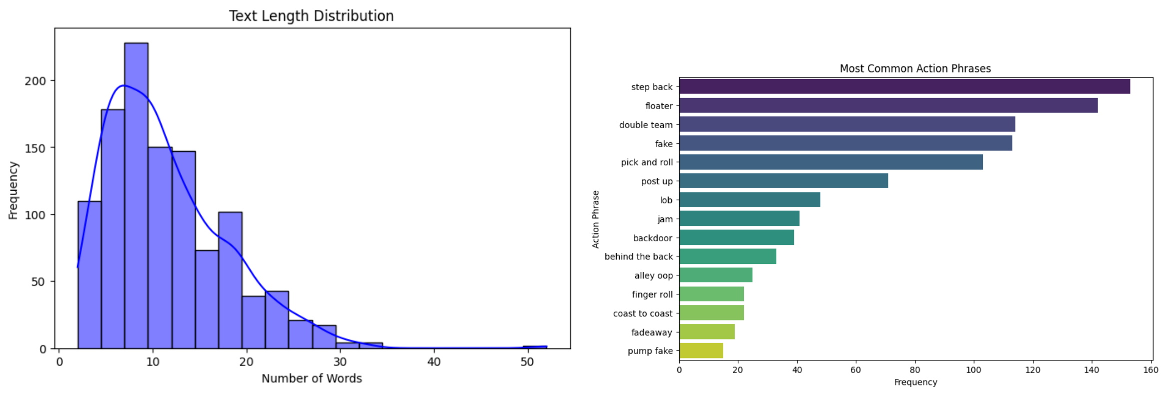

Figure 2: (Left) Text Length Distribution. (Right) Most Common Action Phrases.

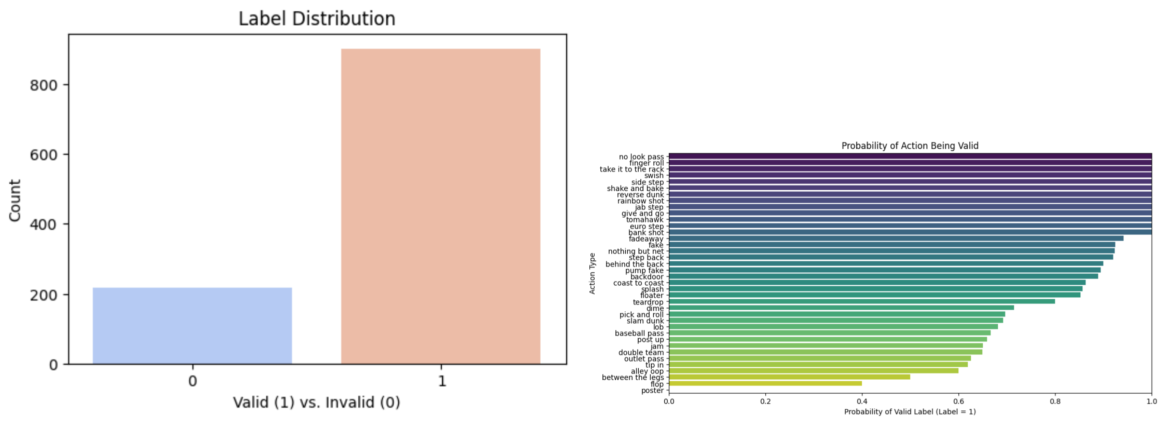

Figure 3: (Left) Label Distribution. (Right) Action Validity Probability.

2.4 Invalid Actions and Action Type Distribution

Figure 4 lists the most frequently misclassified actions, including ‘double team‘ and ‘pick and roll‘. The distribution of all action types, with ‘step back‘ appearing most frequently, is also visualized.

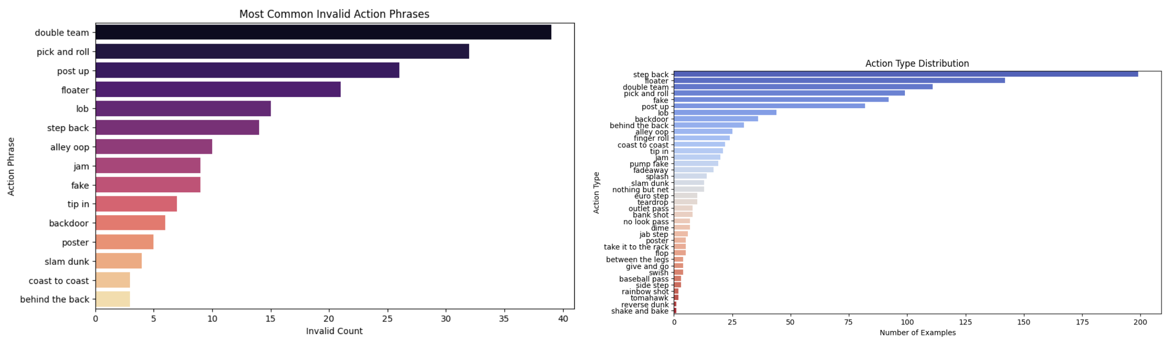

Figure 4: (Left) Most Frequent Invalid Actions. (Right) Action Type Distribution.

3 Modeling Approach

3.1 System Architecture

Figure 5 presents the architecture of the action detection system. The pipeline consists of several key components:

- **Sentence Preprocessor**: This module standardizes the input sentence by applying text normalization, stopword removal, and lemmatization. This ensures consistent representation before action extraction.
- **N-Gram Creator**: Generates n-grams from the input sentence to improve action phrase detection. This step enhances recall by capturing multi-word expressions that match known action phrases.
- **Levenshtein-Based Retrieval**: Uses Levenshtein distance to match input n-grams against the predefined database of action phrases, allowing for minor variations and misspellings.
- **BERT-Based Classifier**: A transformer-based deep learning model that classifies whether an extracted action is valid or not. The classifier captures contextual relationships to disambiguate action relevance.
- **Actions Database**: A predefined repository of known action phrases used for retrieval and classification.
- **Output Module**: Produces the final validated action or an empty result if no valid action is found.

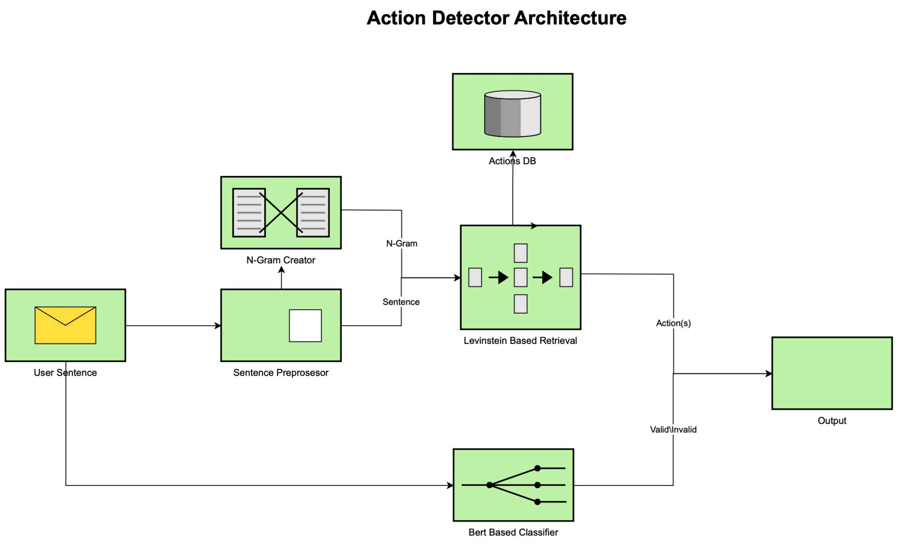

Figure 5: System architecture for action recognition. The process includes preprocessing, n-gram extraction, Levenshtein-based retrieval, and classification using a BERT-based model.

The architecture ensures a robust pipeline that combines linguistic rule-based techniques with deep learning models to maximize precision and recall.

3.2 Sentence Embeddings and Feature Representations

To enhance representation learning, multiple text transformations were applied before embedding generation:

- **Original Sentences** – Unmodified text.
- **Preprocessed Sentences** – Text normalized, stopwords removed, and stemming applied.
- **Contextual Sentences** – Text enriched with action labels, modal verbs, hedging words, and temporal indicators.

A diverse range of vectorization techniques were tested to capture both syntactic and semantic properties of text:

- Traditional methods: Bag-of-Words (BoW), TF-IDF.
- Word embeddings: Word2Vec, GloVe.
- Contextual embeddings: BERT, Sentence Transformers, OpenAI embeddings.

3.3 Comprehensive Experimentation and Evaluation

A large-scale experimentation process was conducted, systematically evaluating different embedding techniques, classifiers, and hyperparameter settings. The evaluation framework included:

- Multiple classifiers: Logistic Regression, SVM, Random Forest, Gradient Boosting, MLP, LSTM, and fine-tuned DistilBERT.
- Hyperparameter tuning using grid search and Bayesian optimization.
- Ablation studies to assess the contribution of preprocessing steps and feature representations.
- Metrics: precision, recall, F1-score, and ROC AUC.
- Misclassification analysis to identify failure cases and potential improvements.

The impact of different preprocessing techniques was thoroughly analyzed to determine the optimal text transformation pipeline. Results showed that transformer-based embeddings, particularly fine-tuned DistilBERT, consistently outperformed traditional models across all metrics.

4 Results and Analysis

Table 1 presents classification performance across different models. Fine-tuning DistilBERT yielded the highest performance, achieving **83%** accuracy, significantly outperforming both traditional machine learning models and other deep learning approaches.

Model	Precision	Recall	F1-score
Logistic Regression	0.76	0.64	0.68
SVM	0.77	0.70	0.73
Random Forest	0.76	0.70	0.72
Gradient Boosting	0.79	0.72	0.75
MLP (Scikit-learn)	0.80	0.76	0.78
Custom LSTM (PyTorch)	0.75	0.41	0.52
Custom MLP (PyTorch)	0.79	0.75	0.77
Fine-tuned DistilBERT	**0.82**	**0.83**	**0.83**

Table 1: Comparison of Model Performance

Figure 6 presents a combined visualization of the evaluation metrics, including precision-recall curve, ROC curve, and confusion matrix.

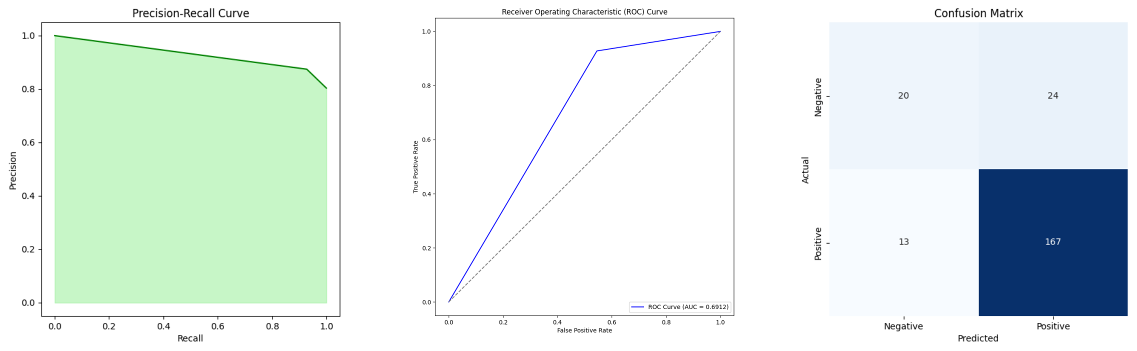

Figure 6: (Left) Precision-Recall Curve, (Middle) ROC Curve, (Right) Confusion Matrix

5 Future Steps

To further enhance the model, three key strategies are proposed. First, context-aware augmentation will modify action phrases, verb tenses, and introduce ambiguity to improve the model's ability to distinguish valid from invalid actions. Second, ensemble models combining classifiers like logistic regression, SVM, and LSTM will enhance robustness across different commentary styles. Finally, Bayesian hyperparameter tuning and a user feedback loop will iteratively refine predictions and improve model accuracy over time.