

Project Update: An NBA Fan AI Agent – Intent Classification and Slot Filling for NBA Statistics Queries

Kevin Zhou
yrzhou@umich.edu

Jiadong Zhu
jiadongz@umich.edu

[GitHub Repository]

1 Introduction

As NBA fans, we often spend significant time searching online for game information and player statistics. Older voice assistants like Siri were limited to basic queries such as scores and match times and could not return detailed data like season averages or individual player stats. Newer systems such as Apple Intelligence and ChatGPT provide richer answers, but they rely on retrieving and synthesizing information from websites, which makes responses slower and less seamless. For fans who want quick, accurate, and user-friendly access to NBA-related data, these solutions remain less than ideal.

This project addresses this gap by building a domain-specific NBA Fan AI Agent that delivers fast and reliable answers to basketball-related queries through natural language understanding (NLU). The core technical challenge lies in jointly performing intent classification and slot filling—mapping a user’s natural-language query to both a high-level intent category (e.g., player statistics, team information) and extracting structured slot values (e.g., player names, attribute types, time frames).

Our approach involves constructing a custom dataset of NBA-related queries, implementing multiple baseline methods for comparison, and training a BERT-based multi-task learning model that jointly predicts intents and slot values. The system is designed to be fast, accurate, and easily extensible to new intent categories or attribute types. We also plan to expand the dataset with more diverse question templates, improve entity extraction through advanced NER or large-language-model (LLM) fine-tuning, and integrate a lightweight retrieval component for handling less common queries.

Solving this problem matters for both technical and practical reasons. From an NLP perspective, it demonstrates how a specialized NLU system can achieve high accuracy with limited, domain-specific data while maintaining interpretability and low latency. From an application standpoint, this project benefits basketball fans, sports media developers, and analytics platforms that require reliable, structured access to player and team information. Moreover, the proposed framework can generalize to other sports or entertainment domains, illustrating how domain-focused NLU can enhance user engagement through efficient, data-driven conversational agents.

2 NLP Task Definition

The core natural language processing (NLP) task in this project is natural language understanding (NLU), which is the combination of intent classification and slot filling. Intent classification maps a user’s query to one of several predefined categories, such as *game schedule*, *player information*, *team information*. Slot filling extracts structured information from the query, such as player names, team names, dates, times, and locations. For example, the query “Which college did Stephen Curry attend?” should be classified under the *player info* intent, while the slots {player: “Stephen Curry”, attribute: “college”} are extracted.

3 Data

3.1 Current Dataset

Our current dataset for the NBA Fan AI Agent was synthetically constructed to support intent classification and slot filling for two intents: *team_info* and *player_info*. All dataset generation scripts are implemented in Python and available in our GitHub repository. The dataset was generated using scripts that define natural-language templates, insert entity names, and automatically an-

notate each example with intent and slot information. Specifically, `build_dataset_team.py` and `build_dataset_player.py` were used to generate the team and player subsets, and `combine_dataset.py` was used to merge, shuffle, and split the final dataset.

We included all 30 NBA teams (e.g., *Warriors*, *Lakers*, *Celtics*, *Bucks*, *Heat*, *Nuggets*, *Mavericks*, *Bulls*, *Knicks*, *76ers*) and 20 well-known players (e.g., *Stephen Curry*, *LeBron James*, *Giannis Antetokounmpo*, *Luka Doncic*, *Kevin Durant*). Each entity was paired with a set of attribute templates describing various factual properties. For teams, the attributes include conference, division, city, full_name, and abbreviation. For players, the attributes include position, height, weight, jersey_number, college, country, draft_year, draft_round, draft_number, and team. For each attribute, three paraphrased question templates were randomly sampled to yield natural-sounding queries such as “Which division are the Warriors in?” or “Which college did Stephen Curry attend?”. Each example is stored as a compact JSON record:

```
{"text": "Which division are the  
Warriors in?",  
"intent": "team_info",  
"slots": {"input": "Warriors",  
"attribute": "division"} }  
  
{"text": "Which college did Stephen  
Curry attend?",  
"intent": "player_info",  
"slots": {"input": "Stephen Curry",  
"attribute": "college"} }
```

After generation, the two subsets (`team.json` and `player.json`) were combined into a single corpus, randomly shuffled, and split into training, validation, and test sets with an 80–10–10 ratio. The final dataset contains roughly 1,050 examples, including about 600 player-related queries and 450 team-related ones. The training set therefore consists of around 840 samples, while the validation and test sets each contain approximately 105 samples. Because all examples were synthetically generated, the intent and slot annotations were derived directly from the templates, ensuring consistent and noise-free supervision for model training and evaluation.

3.2 Future Dataset Improvements

While the current dataset provides strong coverage for player and team queries, it still has several limitations we plan to address. First, we

will introduce a new intent, `game_info`, which will allow the model to handle questions related to NBA games—such as scores, dates, and matchups—extending the agent’s overall capability. Second, the current slot structure only includes a single `input` field, which assumes users always mention the exact name of a team (e.g., “Warriors”) or a player’s full name (e.g., “Stephen Curry”). We plan to expand the slot schema to support more flexible user expressions, such as partial names, nicknames, or contextual references, thereby improving robustness and natural user interaction. Finally, we will increase dataset size and linguistic diversity by enlarging the pool of paraphrased templates—moving from sampling 3 out of 5 per (entity, attribute) pair to selecting 5 out of 10 or more. This will yield a richer and more variable dataset for both training and evaluation.

4 Related Work

The tasks of intent classification and slot filling have been extensively studied in the literature on natural language understanding (NLU), particularly within the context of spoken dialogue systems. Early work often treated the two tasks separately, but more recent approaches emphasize joint modeling for improved performance. Here, we highlight several representative papers that inform our project.

Benchmark datasets such as ATIS, Snips, and CLINC150 have become the standard for evaluating intent classification and slot filling models. ATIS focuses on airline travel queries and remains a long-standing benchmark, Snips covers diverse consumer domains such as music and weather, and CLINC150 provides 150 intents across domains along with out-of-scope queries for OOD evaluation. These datasets can inform the design of our NBA-specific dataset by illustrating how to balance intent coverage, slot schemas, and robustness to domain variation.

[Guo et al. \(2014\)](#) introduced recursive neural networks (RecNNs) for joint intent classification and slot filling on the ATIS corpus. Their model leveraged compositional parse-tree structures to encode hierarchical semantics, outperforming traditional CRF pipelines. However, their approach relied on syntactic parses and domain-specific grammars that do not generalize well to informal sports queries.

[Goo et al. \(2018\)](#) proposed the slot-gated mechanism that explicitly links intent prediction and slot

filling via shared attention weights. Evaluated on ATIS and Snips, it achieved notable gains in joint accuracy and demonstrated that cross-task information flow improves robustness to noisy input. This concept directly informs our architecture: NBA queries often contain ambiguous mentions (e.g., “Heat” vs. “hot”) that benefit from intent-slot interaction.

[Chen et al. \(2019\)](#) fine-tuned BERT for joint NLU, achieving state-of-the-art F1 scores on multiple benchmarks. Their results confirmed that pre-trained contextual encoders capture slot boundaries and rare entities better than sequence-tagging baselines. We adopt this insight by training a multi-task BERT model specialized for NBA data, extending their method to a new, domain-specific dataset.

[Qian and Yu \(2019\)](#) explored meta-learning for domain adaptation in dialogue systems, allowing rapid transfer to new tasks with few labeled examples. Their meta-training strategy motivates our inclusion of out-of-domain (OOD) evaluation, ensuring that our NBA model can gracefully reject unrelated inputs.

Finally, [Larson et al. \(2019\)](#) introduced CLINC150, a large-scale benchmark with 150 intents and OOD detection tests. Their evaluation highlighted the importance of balanced data and explicit OOD categories. We follow their principles in constructing our dataset to maintain intent balance and incorporate unseen query types for realistic evaluation.

Together, these studies reveal that (1) joint models outperform independent ones, (2) pretrained transformers enhance low-resource generalization, and (3) domain adaptation remains critical. Our project builds on these lessons but extends them to a sports-specific NLU setting—a domain with unique linguistic ambiguity (team names, player aliases) and dynamic data requirements. By designing a compact, well-annotated dataset and a lightweight multi-task BERT model, we aim to demonstrate that high-accuracy NLU is achievable even in narrow, rapidly evolving domains such as professional basketball.

5 Methodology

5.1 Current Method

Our current system combines a fine-tuned bert-base-uncased model for intent and attribute prediction with a spaCy-based entity recognizer for extracting player and team names from user

queries. We frame this as a multi-task learning problem, where a single model predicts both the intent (team_info or player_info) and the queried attribute (e.g., height, college, division).

Data Preprocessing: We used the synthetic dataset described in the previous section, combining team.json and player.json into a unified corpus. Each question contains an intent label and an attribute slot. Texts were tokenized using the BertTokenizer with a maximum sequence length of 64, lowercased, and padded to uniform length. Label encoders from scikit-learn converted categorical intent and attribute labels into integer indices. The data was split into training, validation, and test sets using an 80–10–10 ratio, containing approximately 840, 105, and 105 samples, respectively.

Model Architecture: The model (BertForIntentAndAttr) extends a pre-trained BERT encoder with two linear heads—one for intent classification and one for attribute classification. Each head receives the 768-dimensional pooled output from BERT, followed by a dropout layer ($p = 0.2$) and a fully connected layer. The model jointly optimizes the sum of two cross-entropy losses:

$$\mathcal{L} = \mathcal{L}_{intent} + \mathcal{L}_{attribute}.$$

Training Setup: Training was performed on GPU when available, using the AdamW optimizer with a learning rate of 2×10^{-5} , batch size 16, and 5 training epochs. A linear warm-up schedule was applied to stabilize early training. Evaluation used weighted F1-scores and classification reports for both intent and attribute predictions. Model checkpoints and label encoders were saved under models/bert_multi/ for later inference.

Inference: During inference, the text is tokenized and passed through the fine-tuned BERT model to predict the intent and attribute. In parallel, spaCy’s transformer-based NER model (en_core_web_trf) extracts the entity name. Player names are recognized as PERSON, while team names are identified as ORG or GPE. The final output combines both sources, yielding interpretable predictions such as:

Intent: player_info | Attribute: college | Input: Stephen Curry.

Planned Improvements: While this baseline works effectively on our initial synthetic dataset,

it may not generalize well once the dataset becomes larger and more linguistically diverse. As we expand to new intents (e.g., `game_info`) and more varied user expressions, the model may struggle to maintain accuracy. We plan to explore techniques such as continued pretraining, multi-stage fine-tuning, and data augmentation to improve robustness. We will also experiment with lightweight transformer variants (e.g., DistilBERT or RoBERTa-base) to assess performance-efficiency trade-offs.

5.2 Future Methods to Explore

Although the BERT+spaCy pipeline forms a strong starting point, we plan to investigate additional approaches that might improve generalization and scalability.

ChatGPT API: We plan to experiment with OpenAI’s ChatGPT API for zero-shot or few-shot intent and slot extraction through prompt engineering. This approach can potentially handle naturally phrased user queries and unseen entities without retraining. However, prompt consistency and structured output formatting remain open challenges. We will test different prompting strategies (instruction-style, few-shot with examples) and evaluate latency, cost, and stability in large-scale scenarios.

Dialogflow Essentials: We also intend to explore Google Dialogflow Essentials as a complementary framework for production-level intent management. Dialogflow provides built-in tools for intent classification and entity extraction, but at present, we are uncertain how to upload our dataset efficiently, as it currently contains over 1,000 examples and is expected to grow. Further research is needed to determine whether the Dialogflow API supports bulk dataset import and dynamic dataset updates.

In summary, our current pipeline fine-tunes a multi-task BERT model for joint intent–attribute classification and augments it with spaCy-based entity recognition. Future directions include improving this base model to handle a more complex dataset and exploring ChatGPT API and Dialogflow-based systems for enhanced flexibility, robustness, and scalability.

6 Evaluation

We evaluated our models on the held-out test set (10% of the total data, 105 samples) across three

subtasks: (1) intent classification, (2) attribute classification, and (3) input extraction. We compared our main BERT+spaCy pipeline against two baselines—a random predictor and a majority-class predictor—to establish meaningful reference points for model performance and generalization.

6.1 Evaluation Setup

All models were evaluated on the same 80–10–10 train/validation/test split. For the classification tasks (intent and attribute), we report accuracy, macro-averaged F1, and weighted F1 scores. For the input extraction task, we report the exact-match accuracy between the extracted entity and the ground-truth player or team name. Inference time was also recorded to measure runtime efficiency.

6.2 Baselines

We implemented two non-learning baselines:

Random Baseline: randomly assigns intent and attribute labels uniformly across all possible classes and selects a random entity name for input extraction.

Majority-Class Baseline: always predicts the most frequent intent (`player_info`), most frequent attribute (`abbreviation`), and a fixed input entity (*Luka Doncic*) for every query. These simple heuristics provide minimal expected performance levels for comparison.

6.3 Results

Table 1 summarizes test performance across all three subtasks. The random baseline achieved roughly 54% intent accuracy, 2.9% attribute accuracy, and 0.9% input accuracy—consistent with random chance across 2, 15, and over 50 possible outputs, respectively. The majority-class baseline slightly improved intent accuracy to 57.1% and attribute accuracy to 8.6%, but still achieved only 0.9% input accuracy. In contrast, the fine-tuned BERT+spaCy pipeline achieved perfect scores (100%) on both classification subtasks and 78.1% accuracy for input extraction, demonstrating strong performance on template-based queries.

6.4 Inference Speed

Runtime was measured on CPU. The BERT+spaCy model required an average of 6.51 ms per query for BERT inference and 23.27 ms for spaCy NER, totaling 29.78 ms per query. Both baselines executed

Model	Task	Accuracy	Macro F1	Weighted F1	Avg Time (ms)
Random Baseline	Intent	0.543	0.533	0.543	—
Random Baseline	Attribute	0.029	0.026	0.034	—
Random Baseline	Input Extraction	0.009	—	—	—
Majority Baseline	Intent	0.571	0.364	0.416	—
Majority Baseline	Attribute	0.086	0.011	0.014	—
Majority Baseline	Input Extraction	0.009	—	—	—
BERT+spaCy (Ours)	Intent	1.000	1.000	1.000	—
BERT+spaCy (Ours)	Attribute	1.000	1.000	1.000	29.78
BERT+spaCy (Ours)	Input Extraction	0.781	—	—	—

Table 1: Comparison of random, majority, and BERT+spaCy models across all three subtasks on the test set.

nearly instantaneously since they involved no computation. Even so, the main model remains suitable for real-time dialogue applications.

6.5 Discussion

The baselines demonstrate that random and majority heuristics fail to meaningfully capture query semantics, with near-zero performance on attribute and input extraction. The BERT+spaCy model’s perfect classification accuracy confirms it learned the structured template mappings, while its 78.1% input extraction accuracy highlights the limitations of spaCy’s NER model. Although the current synthetic dataset enables near-perfect performance, the model may not generalize to natural, user-generated queries. Future work will evaluate generalization under increased linguistic variability and expanded intent coverage.

6.6 Final Deliverable Plan

For the final deliverable, we plan to integrate multiple intent recognition backends—our finetuned BERT model, the ChatGPT API, and Dialogflow Essentials—with the public [Ball Don’t Lie API](#), which provides up-to-date NBA statistics and player data. This integration will allow the agent to dynamically retrieve real information in response to user queries, such as player stats, team records, or game schedules. We also plan to connect the system to a speech-to-text API to enable natural, voice-based interaction. The final system will thus function as an interactive NBA Fan AI Agent that users can talk to directly, asking questions about players, teams, and games in real time.

7 Work Plan

7.1 Progress and Remaining Timeline

- **Weeks 9–10 (11/3–11/17):** Fine-tune the BERT-based multi-task model on the complete dataset. Conduct systematic hyperpara-

rameter tuning (learning rate, dropout, and sequence length) and evaluate on both dev and test sets using accuracy, macro-F1, and slot-F1 metrics. Begin preliminary integration tests with the `balldontlie` API to verify connection stability and JSON response parsing.

- **Weeks 11–12 (11/18–12/1):** Integrate the fine-tuned NLU model with the `balldontlie` API to enable end-to-end question answering. Perform qualitative error analysis, handle entity edge cases (e.g., ambiguous player names), and compare results with baseline systems. Refine API handling functions for player and team statistics queries.
- **Week 13 (12/2–12/8):** Conduct system-level evaluation focusing on latency, scalability, and runtime efficiency of both model inference and API requests. Prepare visualizations such as confusion matrices, F1 comparisons, and latency histograms. Begin drafting the final poster and report.
- **Week 14 (12/9–12/15):** Finalize the written report and presentation materials. Incorporate instructor feedback, polish documentation, and prepare the GitHub repository for public release, including annotated notebooks for model training and API integration.

7.2 Two-person team justification

We found that dataset creation took longer than expected, primarily due to balancing between attribute coverage and natural phrasing diversity. However, this early investment proved valuable, as it simplified model training and debugging later. Moving forward, we will allocate more time for evaluation and integration, since combining NLU output with live API responses introduces additional edge cases (e.g., player name variants, missing data).

The division of labor remains balanced:

- Team member 1: responsible for fine-tuning and evaluating the BERT model, hyperparameter tuning, and statistical analysis of results.
- Team member 2: responsible for OpenAI API few-shot baseline, integration testing with the balldontlie API, and system latency evaluation.

Both team members will collaborate on dataset validation, evaluation metrics (intent accuracy, macro-F1, slot F1, OOD detection, latency, and cost), error analysis, poster preparation, and the final report. This ensures balanced contributions to the core NLP tasks, while still allowing complementary specialization in different approaches.

References

Qian Chen, Zhu Zhuo, and Wen Wang. 2019. [Bert for joint intent classification and slot filling](#). *Preprint*, arXiv:1902.10909.

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. [Slot-gated modeling for joint slot filling and intent prediction](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 753–757, New Orleans, Louisiana. Association for Computational Linguistics.

Daniel Guo, Gokhan Tur, Wen-tau Yih, and Geoffrey Zweig. 2014. [Joint semantic utterance classification and slot filling with recursive neural networks](#). In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 554–559.

Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. 2019. [An evaluation dataset for intent classification and out-of-scope prediction](#). 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)*.

Kun Qian and Zhou Yu. 2019. [Domain adaptive dialog generation via meta learning](#). *Preprint*, arXiv:1906.03520.