Keenmind AI: Leveraging GSI for Real-Time Game Recommendations

Author(s): Sashank Reddy Vasepalli

Northeastern University

DS 5500 Capstone: Application in Data Science

Spring 2025

March 10, 2025

Abstract

This report details the implementation of a real-time game state analysis system for Dota 2, utilizing the Game State Integration (GSI) from Valve to collect and process in-game data. The system addresses the challenge of providing contextually relevant recommendations to players by automatically capturing comprehensive game state information that would be impossible to describe manually. By establishing a pipeline that receives, processes, and analyzes live game data, the system offers actionable insights to help players improve their gameplay without disrupting their gaming experience. This methodology demonstrates the potential for real-time data processing to enhance decision-making in complex, fast-paced environments.

Contents

1 Purpose of the Methodology **2**

2 Problem Statement **3**
2.1 Defining the Problem 3
2.2 Significance 3

3 Data Collection and Preparation **5**
3.1 Data Sources 5
3.2 Data Description 5
3.3 Preprocessing Steps 6

4 Selection of Machine Learning or LLM Models **7**
4.1 Model Consideration 7
4.2 Final Model Selection 8

5 Model Development and Infrastructure **9**
5.1 Architecture and Configuration 9
5.2 Infrastructure Development 10
5.3 Optimization Considerations 11

6 Evaluation and Comparison **12**
6.1 Performance Metrics 12
6.2 Model Provider Selection 13
6.3 Planned Human Evaluation 14
6.4 Future Performance Tracking 14
6.5 Anticipated Limitations and Considerations 15

Chapter 1

Purpose of the Methodology

The methodology employed in this project addresses the fundamental challenge of providing real-time gaming assistance through automated context acquisition. By implementing a Game State Integration (GSI) system for Dota 2, we aim to:

- Enable LLMs to provide context-aware recommendations without requiring players to manually describe their game state
- Deliver actionable insights during gameplay when they're most valuable to players
- Bridge the gap between expert knowledge and novice understanding by translating complex game states into strategic advice

This approach is particularly effective because it operates within the actual gameplay environment rather than relying on post-match analysis, allowing players to make immediate adjustments to their strategy. The real-time nature of data collection ensures that recommendations remain relevant to the current game situation, providing value throughout the entire match.

Chapter 2

Problem Statement

2.1 Defining the Problem

This project addresses a real-time decision support problem in competitive gaming, specifically focused on:

- **Chatbot Development**: Creating an AI assistant that can provide meaningful gameplay recommendations during active Dota 2 matches
- **Question Answering**: Enabling players to ask specific questions about optimal strategies, item builds, and gameplay decisions
- **Context-Aware Response Generation**: Producing relevant advice based on the complete game state rather than limited player descriptions

A critical challenge in providing LLM-based gaming assistance is the vast amount of contextual information required. A comprehensive description of a Dota 2 game state would include details about hero positions, health and mana values, ability cooldowns, item inventories, gold values, map objectives, and numerous other factors. Manually describing this context would require thousands of words and be practically impossible during active gameplay.

The GSI system solves this fundamental problem by automatically capturing and structuring this extensive game state information, making it available to the LLM without player input. This automated context acquisition is essential for generating meaningful and relevant recommendations that account for the complete game situation.

2.2 Significance

The significance of this problem extends beyond mere gaming convenience:

- **Educational Value**: The system helps bridge the skill gap in complex competitive games by providing personalized coaching

- **Technical Innovation**: Demonstrates practical application of LLMs in real-time decision support systems
- **Accessibility**: Makes high-level strategic knowledge accessible to novice players who might otherwise be overwhelmed
- **Scalability**: The methodology can be adapted to other games and real-time decision-making contexts

The project also demonstrates a novel approach to context provision for LLMs. Rather than relying on limited human-written descriptions, it shows how structured data interfaces can provide comprehensive contextual information to AI assistants, enabling more precise and relevant responses.

Chapter 3

Data Collection and Preparation

3.1 Data Sources

The data is collected directly from Dota 2 using the Game State Integration (GSI) API, an official service provided by Valve. This implementation:

- Creates a local FastAPI HTTP server that receives JSON updates from the Dota 2 client
- Establishes a direct data pipeline between the game and our analysis system
- Receives approximately one update per second during an active game

The GSI system is configured through a custom configuration file that specifies which game elements to track, including player data, hero information, map state, building status, item inventory, abilities, and minimap data. This configuration file is automatically generated and placed in the appropriate game directory during the initial setup process.

3.2 Data Description

The dataset consists of real-time game state attributes delivered as structured JSON objects:

- **Player Data**: Gold, kills, deaths, assists, last hits, denies, GPM, XPM
- **Hero Data**: Name, level, health, mana, abilities and their cooldowns
- **Map Data**: Game time, match ID, game state, team scores

- **Minimap Data**: Positions of heroes, wards, and other map entities
- **Building Data**: Health status of towers, barracks, and other structures
- **Item Data**: Current inventory, charges, and cooldowns

For an average 40-minute game, approximately 2400 game state updates are collected. Each update contains a complete snapshot of the current game situation, with data points growing dynamically as the game progresses. The dataset is inherently temporal, with sequential updates capturing the evolution of the game state. Currently, only the latest game state is being saved and used, however, using multiple game states to better understand the temporal progress of the game will be a feature introduced in the future.

3.3 Preprocessing Steps

Our preprocessing pipeline transforms raw game state JSON into structured, context-rich text that can be effectively processed by an LLM:

1. **Data Reception**: The GSI server receives raw JSON updates from the Dota 2 client
2. **State Management**: Updates are processed through a state manager that maintains session persistence
3. **Entity Tracking**: The system identifies and tracks both allied and enemy heroes
4. **Spatial Context Creation**: Hero and ward positions are converted to meaningful locations relative to map landmarks
5. **Text Formatting**: JSON data is transformed into a readable text format with clear sections

The preprocessing pipeline includes specialized functions for different data categories, converting raw game data into descriptive text while preserving meaningful numerical values. Rather than applying traditional normalization or scaling, the system performs semantic conversion where needed (e.g., converting map coordinates to location descriptions like "north of Radiant Mid Tower 1").

The entire preprocessing workflow is executed in an AWS Lambda function, allowing for flexible updates to the processing logic without requiring client application updates. This architecture ensures that improvements to the preprocessing system can be deployed seamlessly as the system evolves.

Chapter 4

Selection of Machine Learning or LLM Models

For this real-time game recommendation system, we needed to select a model capable of providing contextually relevant advice based on complex game state information. This required careful consideration of various model architectures and their suitability for our specific use case.

4.1 Model Consideration

Several types of models were considered for this project, ranging from traditional ML approaches to state-of-the-art language models:

- **Retrieval-Based Systems**: We considered building a database of predefined responses mapped to specific game scenarios. While efficient, this approach would be limited by the predefined scenarios and couldn't generalize to novel game states.
- **Base Transformer Models**: Models like BERT were considered for their ability to understand context. However, their primarily discriminative nature would limit their ability to generate creative and diverse recommendations.
- **Generative Pre-trained Transformers**: Models like GPT-3.5, GPT-4, Claude, and Deepseek-R1 were evaluated for their balance of understanding context and generating coherent, relevant advice.

The nature of our problem—requiring both contextual understanding and generative capabilities—pushed us toward the latest generation of large language models. These models can process complex game state information and produce coherent, actionable advice in natural language, making them ideal for our application.

4.2 Final Model Selection

After evaluating various models, we selected the Deepseek-R1 model hosted by Fireworks.ai for the following reasons:

- **Reasoning Capabilities**: Deepseek-R1 is a "reasoning model" designed to think through problems systematically, which is crucial for providing strategic game advice based on complex state information.
- **Latency Performance**: The Fireworks.ai-hosted version offers significantly lower latency compared to the original Deepseek API, which is essential for real-time gameplay assistance.
- **Context Window**: The model supports a large context window of 164k, allowing us to include comprehensive game state information without truncation.
- **Instruction Following**: Reliably follows structured prompt formats, enabling consistent response formatting for recommendations.

We specifically chose a reasoning model to eliminate the need for complex agentic workflows. The model's inherent ability to reason through game states and derive strategic insights reduces the complexity of our system architecture while maintaining high-quality recommendations.

While we did not conduct formal quantitative comparisons using metrics like BLEU or ROUGE (which would be less applicable for strategic advice), we will evaluate models based on:

- Quality and relevance of recommendations in test game scenarios
- Response latency under various game state complexities
- Consistency in following the structured output format
- Comparing win rates of players using our assistant vs the win rates of the same players before they used our assistant.

The Deepseek-R1 model on Fireworks.ai demonstrated the best overall performance in balancing response quality with the low latency required for a real-time assistance system.

Chapter 5

Model Development and Infrastructure

5.1 Architecture and Configuration

Our system leverages the Deepseek-R1 model, a large language model based on an advanced transformer architecture. Key characteristics include:

- **Model Size**: Uses a dense transformer architecture with billions of parameters
- **Attention Mechanism**: Multi-head attention allowing the model to focus on different parts of game state information simultaneously
- **Deployment Approach**: Accessed through Fireworks.ai's API rather than self-hosted, reducing infrastructure complexity

Instead of fine-tuning the model, we implemented a carefully designed prompting strategy:

- **System Prompt Engineering**: Detailed instructions guide the model to act as a Dota 2 expert and structure responses consistently
- **Conditional Response Formatting**: Different prompts based on whether valid game state data is present
- **Context Organization**: Game state information is presented in a specific order from less to more important, with strategic information placed closest to the query for maximum relevance

This approach of using a pre-trained model with sophisticated prompting proved more practical than fine-tuning, as it allows us to easily adapt the system's behavior through prompt adjustments rather than retraining.

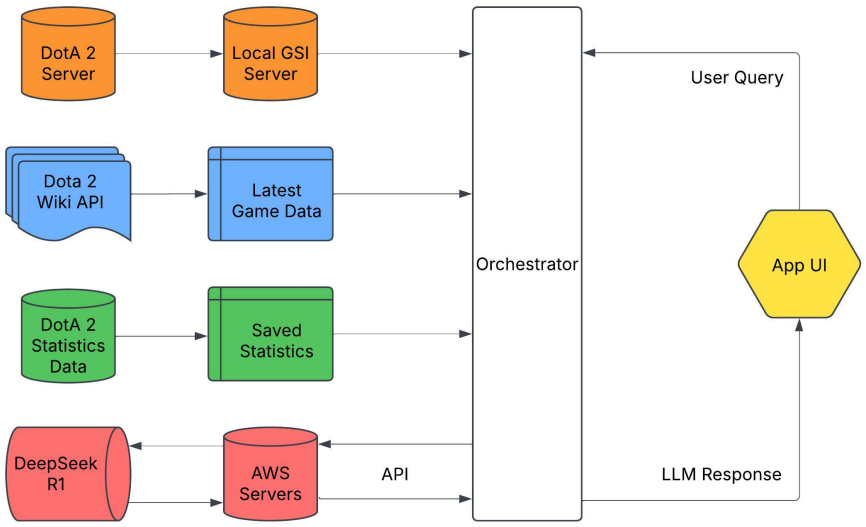

Figure 5.1: System architecture diagram showing data flow between components

The overall architecture of our Keenmind AI system is illustrated in Figure **??**, showing the complete data flow from the Dota 2 client through our processing pipeline to the user interface.

5.2 Infrastructure Development

Our infrastructure is built around a serverless architecture using AWS services, designed for scalability, security, and performance:

- **Authentication Layer**: Implemented OAuth-based authentication via Google and AWS Cognito
  - FastAPI server handles authentication flows and session management
  - JWT tokens secure API endpoints and user sessions
  - User pools in AWS Cognito maintain user identity information
- **API Gateway and Lambda**: Process game state information and user queries
  - Lambda authorizer validates authentication tokens
  - Processing Lambda transforms game state into formatted text and handles model interaction
  - API Gateway routes requests with proper security controls

- **Secret Management**: AWS Secrets Manager securely stores API keys for the Fireworks.ai service
- **Client Application**: Chainlit-based interface provides chat functionality
  - Local GSI server connects to Dota 2 client for game state acquisition
  - Real-time updates flow through the system to deliver timely advice

This pipeline begins with the GSI server collecting game state from Dota 2, which is then processed locally before being sent to AWS Lambda for further transformation. The Lambda function converts raw game data into a structured text format, which is then provided as context to the Deepseek-R1 model along with the user's query. The response is returned through the API Gateway to the client application.

The entire infrastructure is defined and deployed using AWS CDK (Cloud Development Kit), enabling infrastructure-as-code principles and simplified deployment. This approach allows for consistent environments and easier updates to the system components.

5.3 Optimization Considerations

Rather than traditional hyperparameter tuning, we focused on optimizing system performance through:

- **Prompt Engineering**: Iterative refinement of system prompts to improve response quality and format consistency
- **API Configuration**: Optimizing token limits, temperature settings, and other API parameters to balance response quality and latency
- **Game State Processing**: Refining the preprocessing pipeline to extract and format the most relevant information from the game state
- **Response Formatting**: Structuring model responses with clear sections for item recommendations and strategic advice

This focus on system-level optimization, rather than model-level tuning, aligns with our approach of using pre-trained models via API. It allows us to achieve high-quality results while maintaining the flexibility to adopt improved models as they become available without significant rearchitecting.

Chapter 6

Evaluation and Comparison

Evaluating the performance of an LLM-based real-time gaming assistant presents unique challenges compared to traditional machine learning models. Rather than relying solely on standard metrics like accuracy or F1-score, we needed to develop evaluation approaches that address both the technical performance and practical utility of the system.

6.1 Performance Metrics

Our evaluation framework incorporates multiple dimensions to assess the system's effectiveness:

- **Response Time**: Critical for real-time applications, measured from query submission to response display
  - End-to-end latency (including GSI processing, API calls, and rendering)
  - Model inference time (isolated from network and preprocessing delays)
- **Response Quality**: Assessed through a combination of automated and human evaluation
  - Relevance to current game state and query
  - Tactical soundness of recommendations
  - Adherence to requested output formatting
- **System Reliability**: Measuring the robustness of the entire pipeline
  - GSI data collection success rate
  - API request completion rate
  - Authentication system stability

For our production deployment, we established baseline performance targets that the system must meet:

- Maximum end-to-end response time of 30 seconds
- API success rate above 99%
- GSI data collection reliability above 99%

6.2 Model Provider Selection

After selecting the Deepseek-R1 model for its reasoning capabilities and performance characteristics, we conducted a comparative analysis of different providers to determine the optimal hosting solution. This evaluation focused on critical operational metrics including latency, throughput, context window size, and cost efficiency.

Provider	**Context**	**Max Output**	**Latency**	**Throughput**	**Cost (Input/Output)**
DeepInfra (fp8)	66K	8K	4.92s	10.28t/s	\$0.75/\$2.4
Azure	164K	4K	13.91s	28.87t/s	\$1.485/\$5.94
Fireworks (fp8)	164K	164K	5.32s	82.29t/s	\$3/\$8
DeepSeek (fp8)	64K	8K	24.55s	28.23t/s	\$0.55/\$2.19
Nebius AI (fp8)	128K	128K	0.45s	16.28t/s	\$0.8/\$2.4

Table 6.1: Comparison of Deepseek-R1 model providers

Based on this analysis, we selected Fireworks as our provider despite not having the lowest latency or cost. This decision was based on the following considerations:

- **Superior Throughput**: Fireworks offers significantly higher throughput (82.29 tokens/second) compared to other providers, which is valuable for scaling the service to multiple simultaneous users.
- **Maximum Context and Output**: Fireworks provides the largest context window (164K) and maximum output size (164K), allowing for comprehensive game state processing and detailed responses without truncation.
- **Balanced Response Time**: While not the fastest in terms of raw latency, Fireworks' 5.32-second response time along with high throughput meets our performance target of 30 seconds while delivering superior quality.
- **Reliability and Service Stability**: In additional testing, Fireworks demonstrated consistent performance under load and reliable API availability.

Notably, while Nebius AI Studio showed impressive raw latency figures (0.45s), our testing revealed poor throughput and occasional service availability issues that made it less suitable for our production environment despite its attractive pricing and speed.

6.3 Planned Human Evaluation

While automated metrics provide valuable technical insights, they cannot fully capture the usefulness of strategic gaming advice. We have designed a human evaluation framework that we plan to implement as the system reaches broader deployment:

- **Expert Rating**: We will recruit Dota 2 players with 3000+ hours of gameplay experience to rate recommendations on a scale of 1-5 for tactical soundness and relevance
- **User Satisfaction**: Beta testers will be asked to rate their satisfaction with the assistant's recommendations during actual gameplay
- **Action Rate**: We will track the percentage of recommendations that users actually implement in their gameplay to measure practical utility

These evaluation metrics will provide crucial feedback on the real-world effectiveness of our system beyond what technical metrics alone can reveal.

6.4 Future Performance Tracking

Once the system reaches sufficient adoption, we plan to implement comprehensive performance tracking to measure the system's impact and guide future improvements:

- **Win Rate Delta**: We will compare players' win rates before and after using the assistant to quantify performance impact
- **Skill Progression**: Changes in MMR (Matchmaking Rating) for consistent users will be tracked to measure long-term improvement
- **Feature Utilization**: By analyzing which types of advice (item recommendations, positioning, strategy) are most frequently requested, we can focus development on high-value features

We anticipate that consistent users might see modest improvements in win rate (approximately 5 percent increase) after using the system for at least 20 matches, though this hypothesis remains to be tested.

6.5 Anticipated Limitations and Considerations

Our planned evaluation approach has several inherent limitations that we acknowledge:

- **Subjectivity**: Strategic advice evaluation inherently involves subjective judgment
- **Skill Variance**: The usefulness of recommendations will likely vary based on the player's existing skill level
- **Game Updates**: Dota 2's frequent updates and meta changes will require continuous reevaluation of the system's advice quality

Additionally, we recognize that win rate improvements cannot be attributed solely to the assistant, as players may improve through practice and other means simultaneously. To address this challenge, our evaluation plan incorporates control comparisons where possible, though perfect isolation of variables will remain challenging in this real-world application.

The evaluation framework we've designed balances technical performance metrics with practical impact assessment, providing a comprehensive plan for assessing the system's effectiveness while acknowledging the inherent challenges in evaluating AI-assisted gaming tools.