Natural Language Interface for Classic Text Adventures:
Making Zork Understand Human Language

Josiah Panak
CSPB Natural Langauge Processing | Spring 2025
August 15, 2025

Abstract

This project addresses the rigid command syntax problem in classic text-based adventure games by developing a natural language processing system that translates conversational player input into valid game commands. Three parsing approaches were implemented and evaluated: a basic rule-based parser (40% accuracy), a context-aware hybrid parser (32% accuracy), and an enhanced self-contained parser (100% accuracy). The final system successfully maps varied natural language expressions like "How am I doing?" to Zork commands like "diagnose" with significant accuracy. The key finding is that sophisticated parsing without external dependencies outperforms both oversimplified approaches and overengineered context-dependent systems. This work demonstrates that classic games can be made accessible to modern players through careful NLP design that respects grammatical structure while remaining self-contained.

1. Introduction

Classic text-based adventure games like Zork (1980) require players to use exact command syntax with minor variations, which either creates a frustrating barrier between player intent and game interaction, or is simply just less dynamic and immersive then more modern approaches that use pre-written responses. Players must type "go east" or "east" – natural expressions like "walk to the east" are rejected, leading to the infamous "I don't understand that" response from the engine.

This project aims to address this 40-year-old problem by building a natural language understanding system that interprets varied player inputs and maps them to valid Zork commands. The primary research questions are: (1) Can natural language be reliably mapped to fixed command sets? (2) What parsing approach achieves the best accuracy? (3) How should sophistication be balanced with reliability?

The goal is to achieve >90% command accuracy with <100ms response time, enabling players to interact naturally without strictly memorizing command syntax.

2. Related Work

Several researchers have tackled natural language understanding in text-based games. Narasimhan et al. (2015) pioneered using deep reinforcement learning for text games, introducing state representation learning that maps textual descriptions to vector representations. While their focus was on learning to play games rather than parsing commands, their embedding techniques for mapping text to vectors directly informed the semantic similarity matching approach used in all three parsers implemented in this project.

Fulda et al. (2017) introduced affordance extraction via word embeddings, demonstrating how to infer object interactions without explicit programming. Their concept of learning what actions are possible with which objects was implemented in the enhanced parser's affordance system, where objects like "lamp" are associated with actions like "light" and "extinguish." This affordance mapping improved command validation by ensuring suggested actions matched object capabilities.

Ammanabrolu and Riedl (2019) proposed graph-based deep reinforcement learning with action pruning based on game state. Their insight that invalid actions should be filtered based on current game context directly influenced the context-aware parser's design, which attempted to build candidate commands only from objects present in the game state. However, the implementation revealed that over-reliance on context can actually harm accuracy when state information is incomplete or unavailable.

The TextWorld framework by Côté et al. (2019) provides a standardized environment for training NLP agents on text games. While initially considered as a testing platform, the project ultimately chose direct Zork integration instead. TextWorld's approach to defining game mechanics through high-level description languages did influence the structure of the action-to-verb mappings used in the enhanced parser, though the framework itself was not utilized.

3. Data

3.1 Dataset Creation

A custom test dataset of 50+ natural language commands was created, covering 15 action types essential to Zork gameplay:

- **Movement:** "go north", "head east", "I want to walk south"
- **Object manipulation:** "take", "drop", "open", "close", "examine"
- **Complex actions:** "put X in Y", "give X to Y", "unlock X with Y"
- **Meta commands:** "inventory", "save", "look around"

3.2 Structure

Each test case contains:

- Natural language input (e.g., "Could you please pick up that shiny lamp?")
- Expected Zork command (e.g., "take lamp")
- Action type classification
- Confidence threshold

3.3 Preprocessing

Text preprocessing involved:

- Lowercasing and punctuation removal
- Tokenization with special handling for phrasal verbs ("pick up")

- Synonym normalization (lamp/lantern, mailbox/letterbox)
- Filler word identification (but selective removal based on parser version)

3.4 Limitations

The dataset focuses on single commands rather than command sequences. Multi-step instructions like "open the mailbox and take what's inside" are not covered. Additionally, game-specific magic words (e.g., "xyzzy") are excluded as they don't follow natural language patterns.

4. Methodology

4.1 Three-Parser Evolution

Parser 1: Basic Rule-Based (ZorkNLPParser)

- Pattern matching with keyword detection
- Hardcoded synonym dictionaries
- Aggressive filler word removal
- Sentence-BERT embeddings (all-MiniLM-L6-v2) for unknown phrases

Parser 2: Context-Aware Hybrid (ContextAwareZorkNLPParser)

- Context-driven candidate generation from game state
- Pronoun resolution ("it" → last referenced object)
- Weighted embedding matching with presence bonuses
- Required rich game context to function properly

Parser 3: Enhanced Self-Contained (EnhancedNLPParser)

- Sophisticated entity extraction preserving grammatical structure
- Action-specific word removal (only removes relevant words per action)
- Comprehensive verb mappings (20+ action types)
- Affordance system without context dependency

4.2 Technical Implementation

- **Language:** Python 3.8+
- **NLP Libraries:** sentence-transformers, scikit-learn
- **Game Integration:** pexpect for process communication with Frotz
- **Models:** Sentence-BERT (all-MiniLM-L6-v2) for semantic similarity

4.3 Evaluation Metrics

- **Action classification accuracy:** Correct action type identification
- **Command generation accuracy:** Valid Zork command production
- **Response time:** Parse latency in milliseconds
- **Robustness:** Handling of edge cases and unknown inputs

5. Results

5.1 Comparative Performance

Parser	Test Cases Passed	Avg. Response Time
Basic Rule-Based	40%	12.28
Context-Aware	32%	23.98 ms
Enhanced Self-Contained	100%	0.44 ms

5.2 Test Case Analysis

The dramatic accuracy differences stem from fundamental design decisions:

Basic Parser (40%): Failed on complex commands due to over-aggressive word removal:

- Input: "put the lamp in the mailbox"
- After filtering: "put lamp mailbox" (missing preposition)
- Output: INVALID COMMAND

Context-Aware (32%): Failed without proper context:

- Required game state that wasn't available during testing
- Candidate generation produced empty or incorrect sets
- Embedding matching selected wrong commands from poor candidates

Enhanced Parser (100%): Succeeded through intelligent design:

- Input: "put the lamp in the mailbox"
- Preserved structure: verb + object + preposition + container
- Output: "put lamp in mailbox" ✓

5.3 Key Success Factors

The 100% accurate parser succeeded by:

1. **Selective word removal:** Only removing action-specific keywords
2. **Structure preservation:** Maintaining grammatical relationships
3. **Comprehensive coverage:** Handling all Zork command types
4. **Self-containment:** Not requiring external game state

6. Discussion

6.1 Why Context-Aware Failed

The context-aware parser's poor performance (32%) reveals a critical insight: sophisticated features can harm accuracy when their prerequisites aren't met. The parser expected rich game state:

```
game_context = {
    "room_objects": ["lamp", "mailbox"],
    "inventory": ["sword"],
    "npcs": ["troll"]
}
```

Without this context, its complex logic failed catastrophically, performing worse than the simpler baseline. This demonstrates that architectural complexity must be justified by available data.

6.2 The Goldilocks Principle

The results follow a "Goldilocks" pattern:

- **Too simple (40%):** Over-strips words, loses semantic structure
- **Too complex (32%):** Over-engineers, fails without necessary conditions
- **Just right (100%):** Sophisticated parsing, self-contained operation

6.3 Implications for NLP Design

The perfect accuracy of the enhanced parser suggests that for command translation tasks:

1. Preserve grammatical structure rather than aggressively filtering
2. Avoid dependencies on external state unless guaranteed available
3. Use action-specific processing rather than one-size-fits-all approaches

6.4 Limitations

While achieving 100% accuracy on test cases, the system still cannot:

- Resolve pronouns ("take it")

  requires context aware implementation
- Handle multi-step commands

  While I intentionally avoided stringing together commands because it can cause the player to enter irreversible game states before they were given appropriate context, commands like"open the door and go through" should at least be resolved as “open door” but the parser fails to recognize any legal actions from input like this.
- Interpret commands with multiple legal actions referenced

  “look in the mailbox” is incorrectly interpreted as “look” instead of “open the mailbox”

7. Conclusion & Future Work

This project successfully created a natural language interface for Zork that achieves impressive accuracy on standard commands. The key insight is that sophisticated self-contained parsing outperforms both oversimplified and over-engineered approaches.

However, I still believe a context aware approach would outperform my parser in more non-standard cases if it was implemented successfully.

Future Work

1. **Minimal state tracking:** Add lightweight context at least for pronoun resolution
2. **Multi-command parsing:** Handle sequential instructions
3. **Generalization:** Adapt to other text adventures (Zork II, Adventure, etc.)
4. **Learning component:** Fine-tune BERT specifically on game commands

5. **User study:** Evaluate with real players to measure experience improvement

This work demonstrates that bridging natural language with legacy interfaces is achievable through careful parser design that respects linguistic structure while remaining practically grounded.

Bibliography

Ammanabrolu, P., & Riedl, M. O. (2019). Playing Text-Adventure Games with Graph-Based Deep Reinforcement Learning. *Proceedings of NAACL-HLT 2019*, 3557-3565. https://arxiv.org/abs/1812.01628

Côté, M., Kádár, Á., Yuan, X., Kybartas, B., Barnes, T., Fine, E., Moore, J., Hausknecht, M., El Asri, L., Adada, M., Tay, W., & Trischler, A. (2019). TextWorld: A Learning Environment for Text-based Games. *Proceedings of the IJCAI Workshop on Computer Games*, 41-75. https://arxiv.org/abs/1806.11532

Fulda, N., Ricks, D., Murdoch, B., & Wingate, D. (2017). What Can You Do With a Rock? Affordance Extraction via Word Embeddings. *Proceedings of IJCAI 2017*, 1023-1029. https://arxiv.org/abs/1703.03429

Narasimhan, K., Kulkarni, T., & Barzilay, R. (2015). Language Understanding for Text-based Games Using Deep Reinforcement Learning. *Proceedings of EMNLP 2015*, 1-11. https://arxiv.org/abs/1506.08941