Project Title: Word Hunt

An AI-Enhanced Word Search Game with Advanced Features

Submitted By:
Syeda Fakhira Saghir [22k-4413]
Aafreen Mughal [22k-4448]
Muhammad Raza [22k-4499]
Course: AI
Instructor: Sir Khalid Khan
Submission Date: May 2025

1 Introduction

Word Hunt is an AI-powered word search game that dynamically generates puzzles. The game features:

- Three distinct game play modes:
  1. Single-player
  2. Two-player
  3. Human vs AI

- 20 progressive difficulty levels with increasing grid sizes (5x5 to 16x16)
- Thematic word categories with visual backgrounds according to the theme
- Intelligent AI opponent mode with multiple difficulty levels
- Comprehensive scoring system with time bonuses
- Modern UI with theme-based color schemes

2 System Architecture

The game follows a multi-layer architecture:

- **Presentation Layer**: Tkinter-based GUI with theme support
- **Game Logic Layer**: Manages game state, rules, and progression
- **AI Layer**: Implements various search strategies for AI opponent
- **Data Layer**: Handles word storage, retrieval, and processing

3 Word Extraction and Processing

3.1 Word Bank Creation

The game builds its vocabulary through multiple methods:

1. **Initial Word Bank**: Predefined sets of words for basic categories
2. **Web Scraping**: Retrieves additional words from online sources
3. **API Integration**: Uses Merriam-Webster and other dictionary APIs
4. **Heuristic Selection**:
    - Words scored by length (4-8 letters preferred)
    - Letter diversity (unique letters bonus)
    - Common letter frequency (E,T,A,O,I,N bonus)

3.2 Word Similarity Analysis

The game uses TF-IDF vectorization and k-nearest neighbors to:

- Find semantically related words, Generate contextual hints, Enhance AI decision-making

4 Game Flow

4.1 Main Menu Options

- **Game Mode Selection**:
    - Single-player
    - Two-player competitive
    - Human vs AI challenge
- **Theme Selection**:
    - Animals
    - Fruits
    - Countries
    - Random selection
- **Level Progression**: 20 levels with increasing difficulty

4.2 Core Gameplay Loop

1. **Grid Generation**:
   - Words placed horizontally, vertically or diagonally
   - Valid intersections ensured
   - Remaining spaces filled with random letters
2. **Word Finding**:
   - Players can:
     - Click adjacent letters
     - Type words directly
     - Request hints
   - Validation checks:
     - Word exists in grid
     - Letters are properly connected
     - Word not already found
3. **Scoring**:
   - Base points: word length $\times$ 10
   - Time bonus for fast completion
   - Level completion bonus: grid size $\times$ 20
4. **Progression**:
   - Grid size increases every 3 levels
   - AI difficulty adapts to player level
   - New themes unlock with progress

5 AI Implementation

5.1 Search Strategies

Table 1: AI Search Strategy Comparison

Strategy	Time Complexity	Use Case	Difficulty Level
Random	O(1)	Easy Mode	1-2
Longest Word	O(n log n)	Medium Mode	3-4
BFS	O(n)	Hard Mode	5-7
DFS	$O(b^d)$	Hard Mode	8-10
Hill Climbing	$O(n^2)$	Expert Mode	11-15
Minimax	$O(b^d)$	Expert Mode	16-20

Table 2: AI Algorithms and Their Applications

Algorithm	**Purpose**	**Implementation Details**
TF-IDF + KNN	Word similarity	• Vectorizes words • Finds semantic neighbors • Powers hint generation
Breadth-First Search	Word discovery	• Explores all possible words • Guarantees shortest path • Used in Hard difficulty
Depth-First Search	Word discovery	• Explores deep paths first • With depth limit • Used in Hard difficulty
Minimax	Optimal moves	• With alpha-beta pruning • Evaluates word selections • Used in Expert mode
Hill Climbing	Local optimization	• Random restarts • Scores word choices • Balances exploration/exploitation

5.2 AI Algorithms Implementation

5.3 Usage in Game

- **Difficulty Scaling**:
  - *Easy (1-2)*: Random selection (O(1))
  - *Medium (3-4)*: Longest word priority (O(n log n))
  - *Hard (5-7)*: BFS with time limit
  - *Very Hard (8-10)*: DFS with depth limit
  - *Expert (11-20)*: Hill Climbing and Minimax
- **Hint Generation**:
  - Uses KNN to find related words
  - Considers word frequency and position
  - Provides contextual clues (first/last letters)

6 Technical Specifications

- **Programming Language**: Python 3.8+
- **Libraries**:
  - Tkinter (GUI)
  - Scikit-learn (NLP)
  - BeautifulSoup (Web Scraping)
  - Requests (API Calls)

  - Pillow (Image Processing)
  - Numpy (Vector Operations)
- **System Requirements**:
  - 2GHz processor
  - 4GB RAM
  - 50MB disk space
  - Internet connection (for API features)

7 Natural Language Processing (NLP) Integration

7.1 NLP Applications

The game employs NLP techniques in several core functionalities:

1. **Word Similarity Analysis**:
   - Uses `TfidfVectorizer` from scikit-learn to create word embeddings
   - Implements k-nearest neighbors (KNN) to find semantically related words
   - Finds synonyms and related terms for hints
2. **Web Scraping for Vocabulary Expansion**:
   - Parses HTML from dictionary sites using BeautifulSoup
   - Extracts synonyms and related terms contextually
   - Filters words by length and validity for game use
3. **Word Placement Heuristics**:
   - Scores words based on letter frequency patterns
   - Prioritizes words with common prefixes/suffixes
4. **Hint Generation**:
   - Provides word definitions as contextual clues
   - Shows first/last letters with length indicators
5. **AI Decision Making**:
   - Uses word frequency statistics
   - Considers letter distribution patterns

7.2 Limitations

The implementation has some NLP constraints:

- No deep semantic analysis (word2vec/GPT-style understanding)
- Limited contextual awareness beyond basic similarity
- Dependency on pre-scraped word banks for performance

Table 3: NLP Components vs Traditional Programming

Feature	**NLP Technique Used**	**Traditional Alternative**
Word Similarity	TF-IDF + KNN	Hardcoded synonym lists
Hint Generation	Contextual analysis	Random word selection
AI Word Choice	Frequency analysis	Pure random selection

8 Visual Design and UI

8.1 Theme System

The game features a comprehensive theme system with:

- **Category-Specific Colors**: Different color schemes for each word category
- **Dynamic Backgrounds**: Pexels API integration for thematic backgrounds
- **Image Processing**:
  - Gaussian blur for readability
  - Brightness adjustment
  - Resizing and cropping

Conclusion

Word Hunt successfully demonstrates:

- Practical NLP applications in game design
- Effective AI opponent implementation with multiple strategies
- Engaging progressive difficulty system
- Robust word processing pipeline with web scraping
- Modern UI with theme support and visual effects

The project showcases how traditional word games can be enhanced with modern AI techniques while maintaining accessibility and fun gameplay. Future work could expand NLP capabilities and add more interactive features.

References

[1] Scikit-learn: Machine Learning in Python, Pedregosa et al.

[2] Tkinter GUI Application Development, Bhaskar Chaudhary

[3] Speech and Language Processing, Jurafsky & Martin