Overview:

The Goal of our project is to develop an AI-based solver for the New York Times Strands puzzle utilizing some combination of the AI techniques we've learned in lecture. Strands is a word puzzle game that requires players to identify words from a letter grid, following specific patterns, while also identifying a hidden "spangram", a word using letters that span the grid. We plan to build a solver that aims to automate the solving process by leveraging probabilistic modeling techniques, to identify likely word sequences and improve accuracy through Reinforcement Learning.

Problem Statement:

To solve NYT Strands, you need to employ a combination of pattern recognition, word association, and search optimization. This makes it an ideal candidate for us to streamline and identify its solution with AI. The key challenge is efficiently navigating the grid while considering the context of the theme and length of word choice.

Existing approaches:

1. https://github.com/hrfee/filaments
2. https://arxiv.org/abs/2404.11730
3. https://engineering.nyu.edu

Unlike the Filaments project, which focuses on multiplayer gameplay rather than automated solving, our method provides an AI-driven solution that efficiently identifies optimal word paths. Compared to the LLM-based approach in the arXiv paper, which relies on semantic understanding and categorical grouping, our model explicitly captures the structural dependencies of Strands puzzles, ensuring a more precise and computationally efficient solution. Additionally, our adaptive learning mechanisms dynamically refine transition probabilities, making our solver more robust and scalable.

Proposed Approach:

Our proposed approach extends traditional Hidden Markov Models by incorporating:

1. Custom Transition Probabilities: Adapting Hidden Markov Models to model letter-to-word transitions.
2. Grid-Based Path Dependencies: Enhancing Hidden Markov Models to work with spatial constraints in the puzzle.
3. Reinforcement Learning: Using reinforcement learning techniques to refine transition probabilities dynamically.
4. Graph Algorithms: Use BFS/DFS to identify words with potential for optimization with heuristics.