Generative Crossword Construction with Reinforcement Learning

A proof-of-concept pipeline combining large language models, constraint-aware search, and reinforcement learning

Steven Blank | Project completed May 2025

Abstract. Crossword construction combines open-ended language generation with a tightly constrained combinatorial search problem. This project develops a proof-of-concept system that accepts a user-supplied theme and produces a filled crossword grid with associated clues. The pipeline has three stages: a large language model proposes and scores thematic entries; a constraint-aware Q-learning and search procedure fills a grid while balancing completion, fill quality, and theme relevance; and a hybrid clue module retrieves clues from a large database with an LLM fallback. In the reported baseline experiment, the Q-learning configuration completed the test grid while achieving a higher average word-quality score than a random-action baseline, at the cost of additional runtime. Small-scale parameter experiments also suggest that prioritizing common, high-quality fill can improve completion rather than merely constrain it. The results establish the feasibility of the integrated pipeline while leaving important questions about robustness, evaluation scale, clue quality, and the contribution of learning relative to heuristics and backtracking.

Project at a glance

Input	A theme such as "space exploration" and a crossword grid template.
Output	A filled grid, themed entries where feasible, and Across/Down clues.
Core idea	Combine LLM-based linguistic generation with a hybrid learned-and-heuristic constraint solver.
Evidence	Complete unthemed fills in the reported setup; improved word-quality score over a random-action baseline; complete or near-complete themed fills in proof-of-concept runs.
Scope	An exploratory research prototype, not a production-ready constructor or a substitute for expert human editing.

1. Research question and contributions

Research question. Can a system combine modern language models with reinforcement learning and constraint-aware search to generate a complete themed crossword puzzle with limited human intervention?

The project's main contributions are:

- an end-to-end pipeline connecting thematic entry generation, grid filling, clue generation, and HTML rendering;
- a grid-filling objective that explicitly balances completion, word quality, theme relevance, and future constraint pressure;
- an empirical comparison between a Q-learning configuration and a random-action baseline, together with exploratory studies of word-quality and theme rewards; and
- a transparent account of the system's limitations, including its reliance on heuristics and backtracking, small evaluation scale, and incomplete clue-generation robustness.

2. System design

The system is organized as a three-stage pipeline. Each stage solves a different part of the construction problem and passes structured outputs to the next component.

1. Theme generation
GPT-4o proposes candidate entries and relevance scores.

→

2. Grid filling
Q-learning-style values, hard constraints, heuristics, and backtracking.

→

3. Clue generation
Clue database first; LLM fallback for missing entries.

2.1 Themed-entry generation

A structured prompt asks GPT-4o for candidate entries subject to word-length and output-format constraints. The response is parsed into entries, definitions, and theme-relevance scores. Candidates that violate grid constraints or do not appear in the accepted crossword lexicon are filtered. Batching produces a broad candidate set so that the grid solver can trade off thematic relevance against fill feasibility.

2.2 Grid filling as a hybrid learning-and-search problem

The grid-filling component builds on Bentivegna's public tabular Q-learning implementation [6], which served as a baseline for the search environment. I substantially extended and integrated that foundation: I redesigned the state features and reward structure, incorporated scored fill quality and theme-specific objectives, added constraint-density features and parameterized experiments, and connected the solver to the theme-generation, clue-generation, and rendering stages. The implementation in this work is best understood as a hybrid system in which learned state-action values guide decisions while hard letter constraints, heuristic entry ordering, and backtracking preserve feasibility.



Figure 1: Example 13×13 grid and clue rendering produced by the pipeline. The artifact demonstrates end-to-end integration; clue and fill quality still require human evaluation.

2.3 Clue generation and rendering

For each filled entry, the system first searches a database of more than five million historical clue–answer pairs. If no suitable clue is found—especially for generated thematic entries—the pipeline can request an LLM-generated clue. The final grid and Across/Down lists are rendered as HTML.

3. Reinforcement-learning formulation

3.1 State and action representation

The state representation includes the current pattern of filled and unfilled cells, the most recent placement, constraint density for remaining slots, and the fraction of the grid completed. An action selects both the next slot and a compatible word. In practice, the candidate action set is pruned by letter compatibility and other crossword constraints before value-based selection.

3.2 Reward design

The reward combines four terms:

$$R = R_{\text{completion}} + R_{\text{quality}} + R_{\text{theme}} - P_{\text{constraint}}. \tag{1}$$

Completion rewards encourage valid progress; word-quality rewards favor common, constructor-approved entries from a scored lexicon; theme rewards favor entries related to the prompt; and the constraint penalty discourages choices that sharply reduce future fill options. The agent applies the standard Q-learning update

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]. \tag{2}$$

Learning rate and exploration were varied during development. Backtracking removes placements when no compatible continuation remains. Because these search heuristics materially contribute to success, the experiments should be read as evaluating the complete hybrid procedure rather than isolating a pure RL agent.

4. Preliminary evaluation

The experiments were exploratory and used a 13×13 grid to reduce runtime. They test whether the integrated pipeline can produce valid fills and whether reward design changes observed fill quality. Due to the constraint of the scale of this study, the results should be treated as proof-of-concept evidence rather than statistically conclusive performance estimates across grids, seeds, or model families.

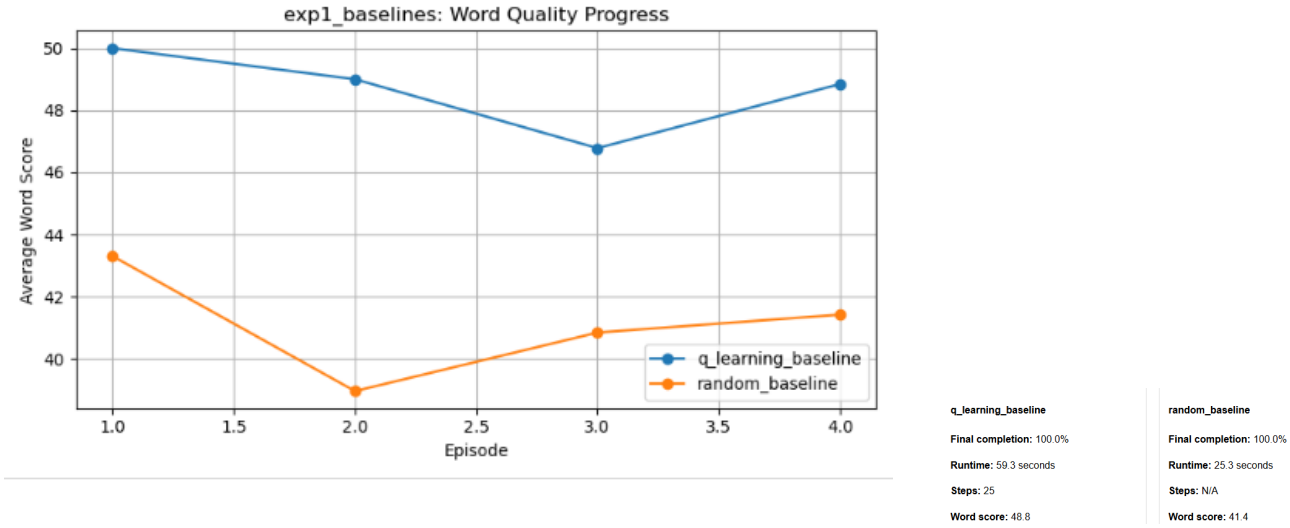

Figure 2: Reported Q-learning versus random-action baseline. Both completed the test grid; the Q-learning configuration produced a higher word-quality score but took longer.

Table 1: Reported baseline comparison.

Configuration	Completion	Word score	Runtime
Q-learning baseline	100.0%	48.8	59.3 s
Random-action baseline	100.0%	41.4	25.3 s

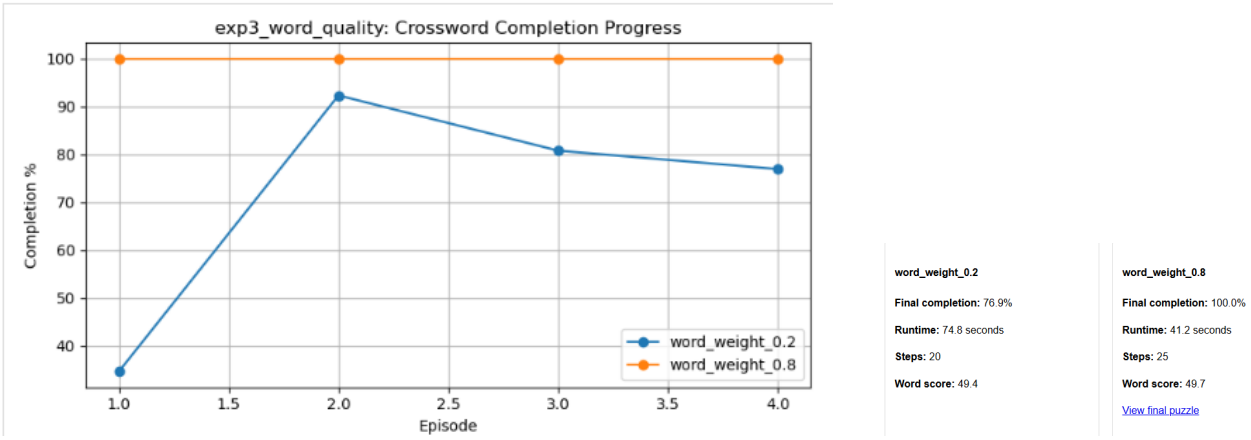

Figure 3: Illustrative comparison of two word-quality reward weights. In this small run, the higher weight coincided with better completion and similar fill quality.

5. Limitations and future work

- **Evaluation scale.** These experiments use a small number of runs and could be built on to include multiple random seeds and a train/test split across unseen grid templates.
- **Attribution.** The current implementation mixes learned values with hand-designed pruning, entry-ordering heuristics, and backtracking. Ablations are needed to measure the marginal value of each component.

- **Puzzle quality.** Automatic scores are imperfect proxies. Expert constructors or experienced solvers should rate fill naturalness, theme coherence, clue fairness, and overall enjoyment.
- **Robustness.** The theme and clue modules require more reliable validation, and the interface needs cleaner error handling and reproducible experiment metadata.
- **Scale.** Future work should test larger grid sizes like the 21×21 Sunday puzzle grids, multiple layouts and difficulty levels, larger theme sets, and computationally stronger search or learned policies.

6. Conclusion

This work demonstrates a technically ambitious proof of concept: a user can provide a theme, an LLM can propose relevant material, and a hybrid RL/search procedure can attempt to assemble a constrained grid before a clue module renders a playable artifact. The end-to-end system also makes the central research challenge visible. Generating plausible language is comparatively easy; producing robust, high-quality constructions depends on search, evaluation, and carefully designed objectives. Crosswords therefore provide a concrete testbed for studying how learned policies interact with hard constraints and imperfect quality proxies—questions that extend well beyond puzzle construction.

Supporting materials

This brief summarizes the system architecture and the results that are most interpretable from the original development work. The codebase, demo video, setup notes, and additional exploratory artifacts are available upon request.

References

[1] L. Rigutini et al., "A Fully Automatic Crossword Generator," *ICMLA*, 2008.

[2] L. Rigutini et al., "Automatic Generation of Crossword Puzzles," *International Journal on Artificial Intelligence Tools*, 2012.

[3] V. Bulitko and A. Eaton, "Evolving Romanian Crossword Puzzles with Deep Learning and Heuristic Search," 2021.

[4] K. Zeinalipour et al., "Building Bridges of Knowledge: Innovating Education with Automated Crossword Generation," *ICMLA*, 2023.

[5] K. Zeinalipour et al., "From Arabic Text to Puzzles: LLM-Driven Development of Arabic Educational Crosswords," 2025.

[6] M. Bentivegna, "Crossword Puzzle Generator," GitHub repository, 2022.

[7] D. Silver et al., "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm," arXiv:1712.01815, 2017.

[8] C. Jones, "Crossword-Wordlist: Scored Wordlist for Use with Construction Software," GitHub repository, 2020.