Solving Crossword Puzzles using Loopy Belief Propagation and SweepClip

Shivam Mittal*
International Institute of Information Technology
Hyderabad, India

Chirag Goyal*
International Institute of Information Technology
Hyderabad, India

Aditya Garg*
International Institute of Information Technology
Hyderabad, India

ABSTRACT

Solving crossword puzzles programmatically is a challenging task that involves understanding clues, enforcing structural constraints, and ensuring semantic coherence. In this project, we propose a hybrid approach that combines the semantic reasoning power of a Bi-Encoder with the structural accuracy of Loopy Belief Propagation (LBP) and the pruning strategy of SweepClip. This paper describes our methodology, algorithms used, and the results obtained. SweepClip showed a substantial improvement in solving accuracy over traditional LBP, validating our approach.

1 INTRODUCTION

Solving crossword puzzles with automation combines two difficult tasks — natural language understanding and structured constraint satisfaction. While large language models (LLMs) can handle free-form reasoning and word associations, they lack the rigor needed for grid-based constraints like intersecting words and fixed lengths.

To address this, we built a hybrid system using:

- **SweepClip:** A method to generate and prune candidate answers iteratively based on grid constraints.
- **Loopy Belief Propagation (LBP):** A message-passing algorithm on a graphical model representing the crossword grid.

These models together help us progressively narrow down the valid words at each grid location while maintaining semantic accuracy.

2 BRIEF OVERVIEW

Large Language Models (LLMs) such as GPT-4 Turbo have demonstrated remarkable capabilities in language understanding and generation. However, solving crossword puzzles imposes unique structural and semantic constraints that standard language tasks typically do not involve. One key limitation of LLMs in this setting is their difficulty in strictly adhering to character length and positional constraints required for crossword entries.

Initial evaluations involved prompting LLMs to generate answers solely based on textual clues, with no additional structural information. In this setup, models frequently produced semantically plausible but length-inconsistent or grid-incompatible answers. The word accuracy in these cases was noticeably suboptimal, reaffirming the need for additional contextual grounding.

To address this, we explored whether LLMs could exploit partial information from a crossword grid — such as already-filled letters from intersecting answers — to improve performance. When presented with these character constraints as part of the prompt, models like GPT-4 Turbo showed significantly better results. This supports the hypothesis that LLMs benefit from structural guidance, not just semantic context.

Empirical results across datasets such as NYT, Cryptonite, and word-init-disjoint indicate that model accuracy improves notably when partially filled grids are provided. For instance, on the word-init-disjoint dataset with 5-shot prompts, GPT-4 Turbo outperforms the previous state-of-the-art (a fine-tuned Mistral 7B model by Sadallah et al., 2024) by approximately 2.8×.

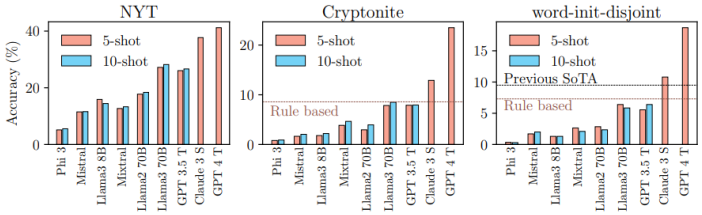

Figure 1: Performance of GPT-4 Turbo on crossword datasets shows 2× improvement over previous SoTA using 5-shot prompts.

Hint (%)	0%		25%		50%		70%
Model	NYT	*init*	NYT	*init*	NYT	*init*	*init*
Mistral 7B (5-shot)	10.95%	1.70%	9.70%	2.80%	11.95%	4.80%	
LlaMa 3 8B (5-shot)	15.8%	1.30%	19.7%	2.85%	24.65%	6.25%	
LlaMa 3 70B (5-shot)	27.20%	6.40%	31.80%	11.45%	45.30%	20.35%	
GPT 4 Turbo (5-shot)	41.2%	18.70%	59.95%	33.70%	75.75%	52.85%	**76.30%**
Mistral 7B† (fine-tuned)							27.0%

Figure 2: LLMs, particularly GPT-4 Turbo, leverage partial grid constraints to boost accuracy, outperforming fine-tuned Mistral 7B by 2.8×.

3 PROJECT IDEA

Crossword puzzles require matching answers to clues under tight constraints (length, position, and overlap with other answers). LLMs provide good guesses, but are not constraint-aware. Hence, we developed an algorithmic approach to complement the LLM's clue-answer generation using grid logic.

4 OUR APPROACH

We propose a two-part system:

- **Bi-Encoder:** Pretrained model (`paraphrase-MiniLM-L6-v2`) scores candidate answers against a clue using semantic similarity.

*All authors contributed equally to this project.

- **Constraint Engine:** Enforces letter overlap, length consistency, and position constraints using LBP or SweepClip.

5 BI-ENCODER FOR SEMANTIC SCORING

We have used the pretrained **Bi-Encoder model (paraphrase-MiniLM-L6-v2)** to evaluate how semantically similar each candidate answer is to a crossword clue.

How It Works

- Generate embeddings for the clue (E_c) and each candidate answer (E_a).
- Compute **semantic similarity score (SS)** using the **dot product** of embeddings.
- Apply **softmax** to get probability scores for each candidate being the correct answer.

Formulas

$$SS_i = E_a \cdot E_c \tag{1}$$

$$p_i = \frac{\exp(SS_i)}{\sum_j \exp(SS_j)} \tag{2}$$

Outcome

Each candidate is assigned a probability score based on how likely it is to be the answer, helping guide grid filling decisions.

6 CROSSWORD SOLVING WITH LOOPY BELIEF PROPAGATION (LBP)

We have used **Loopy Belief Propagation (LBP)** to fill the crossword grid by combining probabilistic candidate answers with grid constraints.

Key Concepts

- The grid is modeled as a **probabilistic graphical model**.
- Each cell is a **VariableNode**, representing 26 letter possibilities (A–Z).
- Each clue is a **FactorNode**, which connects to **VariableNodes** and sends probabilistic messages.

Implementation

- Built using `VariableNode`, `FactorNode`, and CrosswordSolvingGrid classes.
- 25 iterations of LBP are run. Final letter is chosen greedily using highest marginal probability.

Message Passing (LBP)

Messages $m_{i \to j}$ represent a node's belief about a neighbor's letter. These messages are updated using:

$$m_{i \to j} = \sum_{x_i} \left(\psi(x_i)\phi(x_i, x_j)\xi(x_i, x_j) \prod_{k \in \text{neighbors of } i \setminus j} m_{k \to i} \right) \tag{3}$$

$$p(x_i) = \psi(x_i) \cdot \prod_k m_{k \to i} \tag{4}$$

Potential Functions

- $\psi(x_i)$: Prior from Bi-Encoder
- $\phi(x_i, x_j)$: Indicator for across/down match
- $\xi(x_i, x_j)$: Bigram probability for adjacent letters

7 SWEEP CLIP ALGORITHM

The **Sweep Clip algorithm** is typically used in optimization and graph-based tasks, where a set of constraints or variables are iteratively adjusted to reach a feasible or optimal solution. The solver is essentially performing a **sweeping process**, gradually refining its search space and adjusting answers at each step, where "clip" refers to cutting down or pruning incorrect or inconsistent answers.

7.1 Candidate Answer Generation

Initially, the candidate answers are generated by passing the clue and the answer length, in a particular format to an OpenAI model (GPT-4 Turbo).

7.2 Depth = Iteration Step

At each *depth*, the solver refines its guesses, gradually improving the crossword state.

7.3 Constraint Pruning

Answers violating clue constraints are pruned, ensuring consistency across the board.

7.4 Graph-Based Search

A graph of clues is used to track relationships. Edges represent shared letters or overlaps.

7.5 Progressive Refinement

Solver sweeps deeper (increasing depth) until all clues are valid or `max_depth` is reached.

8 RESULTS AND OBSERVATIONS

The **Loopy Belief Propagation (LBP)** algorithm has long been a widely used approach for probabilistic reasoning in crossword puzzle solving. In our implementation, it combined semantic clue information with grid-based constraints through iterative message passing. We observed that LBP typically achieved an accuracy in the range of **60-80%**, aligning with benchmarks reported in earlier studies.

While effective, LBP had limitations in handling complex grid interactions and pruning inconsistent answers. To overcome these, we introduced the **Sweep Clip** algorithm, which brought a significant improvement in performance. By iteratively generating and refining candidate answers while enforcing structural consistency, Sweep Clip achieved a much higher accuracy of approximately **90–95%**.

This performance boost was consistently observed across different datasets and validated through our own experiments. The next sections present a detailed comparison, illustrating the accuracy gains and solving efficiency achieved with the Sweep Clip algorithm.

8.1 Results from LBP Algorithm

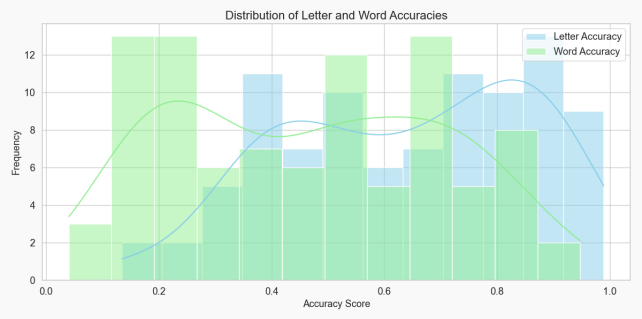

Figure 3: Distribution of Letter and Word Accuracy

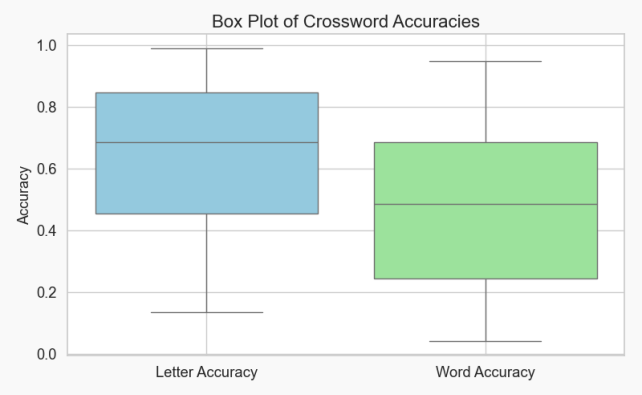

Figure 4: Box Plot of Crossword Accuracies

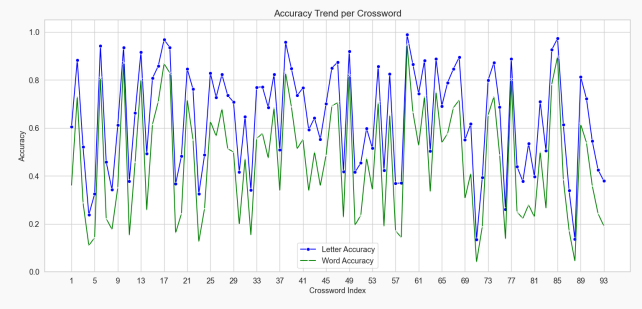

Figure 5: Accuracy test per Crossw

8.2 Results from Sweep Clip Algorithm

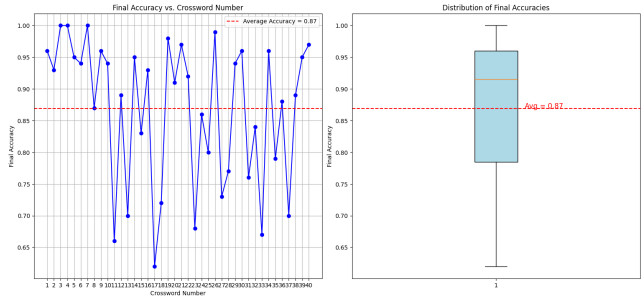

Figure 6: Placeholder: Sweep Clip Result Example 1

REFERENCES

- https://arxiv.org/pdf/2205.09665
- https://web.stanford.edu/class/cs224n/final-reports/256725260.pdf
- https://platform.openai.com/docs/guides/fine-tuning