# The GridFather: Automated Crossword Solving from Images

Team Name: **The Tokenizers**
**Sanchit Kumar** 2024201042
**Anubhav Mishra** 2024201001
**Utkarsh Sachan** 2024201012

Under the guidance of:
**Dr. Manish Srivastava**

**Abstract**

Crossword puzzles combine semantic interpretation, domain knowledge, and spatial reasoning, presenting a rich challenge for natural language processing. In this project, we introduce *The GridFather*, an end-to-end system that solves crossword puzzles from input images. Our pipeline begins with image preprocessing using contour detection to extract the grid, followed by vision based LLMs(GPTs, Florence) to retrieve clues. We explored multiple transformer-based models to generate answer candidates for each clue, including BERT (closed-book), fine-tuned MiniBERT, QWEN, and T5-small. Among these, BERT-base-uncased achieved the highest clue-answer matching performance with a 49% top-500 accuracy, while MiniBERT offered a lightweight alternative at 39.5%. QWEN and T5-small were less effective, achieving top-10 accuracies of 14.55% and 15%, respectively. Candidate answers were ranked using a bi-encoder trained for semantic similarity, and final grid inference was performed using loopy belief propagation to resolve spatial constraints. This report presents our system design, model evaluation based on top-$k$ candidate accuracy, and insights into the practical challenges of solving crosswords from real-world visual inputs.

## 1 Introduction

Crossword puzzles are a popular intellectual exercise, demanding a blend of linguistic dexterity, factual knowledge, and logical reasoning. The New York Times crossword, a daily staple for millions, typically features a 15×15 grid with intersecting clues that vary in difficulty across the week. Traditional solvers rely on human intuition, but recent advances in natural language processing (NLP) have enabled automated solvers. Our project, *The GridFather*, pushes this frontier by accepting an image of a crossword puzzle as input and producing a solved puzzle image as output. This end-to-end system combines image processing, clue interpretation, candidate generation, and grid filling, leveraging transformer models and probabilistic algorithms. We aimed to create a robust, generalizable solver capable of handling diverse clue types, including trivia, wordplay, and metaphors.

## 2 Dataset and Exploration

To train and evaluate our crossword-solving pipeline, we used two primary datasets: the large-scale `CrosswordQA` dataset from Hugging Face for training language models, and a smaller, curated dataset from Kaggle for focused evaluation.

## 1. CrosswordQA Dataset (Hugging Face)

The `albertxu/CrosswordQA` dataset[1] contains over 6.4 million question-answer (clue-answer) pairs extracted from a variety of crossword publishers including The New York Times, LA Times, The Guardian, and more. It spans clues from 1950 to 2021, covering a wide range of trivia, wordplay, and definition-style clues.

**Key Statistics:**

- **Total QA pairs:** 6,403,271
- **Unique answers:** ∼438,000
- **Clue types:** Includes knowledge-based, definitions, commonsense reasoning, and wordplay
- **Answer lengths:** Ranges from 2 to 34 characters, with 5–8 being most common
- **Publisher diversity:** Clues drawn from 26 sources (NYT, LAT, Newsday, The Guardian, etc.)

This dataset was used to fine-tune various transformer-based models (e.g., MiniBERT, QWEN, T5) for clue-answer generation. Its breadth makes it ideal for pretraining models to handle a wide range of clue formulations and difficulty levels.

## 2. New York Times Clues and Answers (Kaggle)

The second dataset[2] is a CSV file consisting of 781,573 crossword clues and corresponding answers sourced exclusively from The New York Times between 1993 and 2021.

**Dataset Format:**

- Columns: `ID, Date, Clue, Answer`
- Clues are plain-text strings (e.g., "Capital of Norway")
- Answers are single words or multi-word strings with no spaces (e.g., "OSLO", "ALLRISE")
- Dates span 28 years, allowing for temporal analysis

**Use in Project:**

- Used primarily for validation and small-scale fine-tuning
- Sampled to evaluate top-$k$ clue-answer matching performance
- Allows for focused testing on NYT-specific clue styles and themes

## 3. Observations and Insights

- **Answer Distribution:** Short answers (3–6 letters) dominate both datasets, aligning well with crossword conventions.
- **Clue Diversity:** The CrosswordQA dataset offers richer variation in clue types, while the NYT dataset has more consistent formatting.
- **Temporal Drift:** Many clues reference pop culture or current events, which could cause models to perform better on clues from years closer to their training corpus.

[1] `https://huggingface.co/datasets/albertxu/CrosswordQA`
[2] `https://www.kaggle.com/datasets/darinhawley/new-york-times-crossword-clues-answers-19932021`

- **Data Cleaning:** Minimal cleaning was required due to the datasets' structured formatting, but lowercasing and punctuation handling were standardized.

## 4. Temporal and Weekly Accuracy Trends

We also analyzed the average letter prediction accuracy (based on candidate ranking performance) across years and days of the week to evaluate systematic trends and difficulty levels in the NYT dataset.

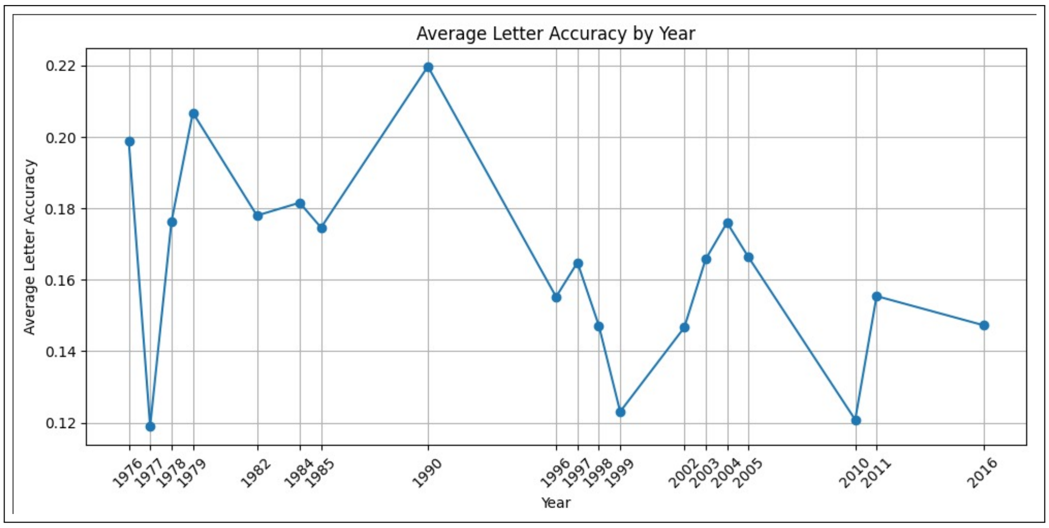


Figure 1: Average letter accuracy by publication year. Variability suggests temporal drift and evolving clue style difficulty.

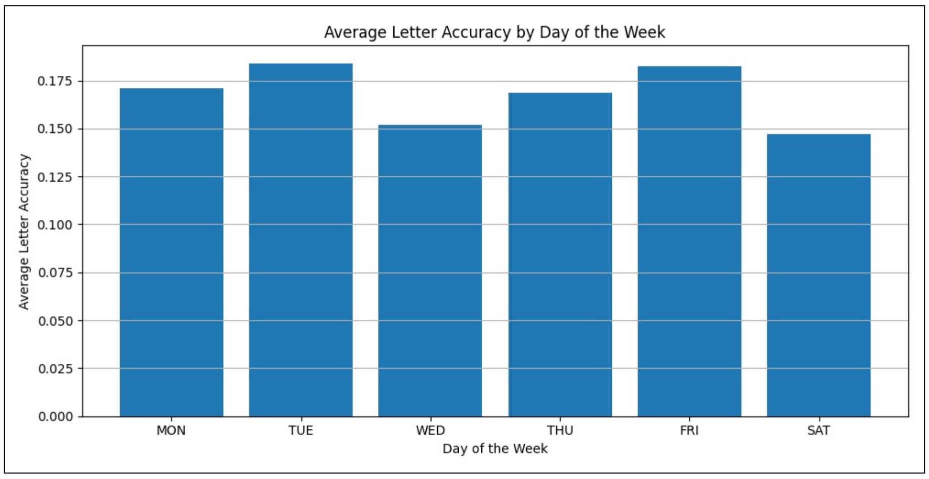


Figure 2: Average letter accuracy by day of the week.

These plots show that older and weekend puzzles tend to present more difficulty for the model, likely due to a combination of complex wordplay, rare trivia, and looser clue formatting.

## 3 Methodology

Our pipeline consists of four primary stages, each contributing a critical component to the end-to-end automated crossword solver:

- **Image Processing:** Detect the crossword grid using computer vision techniques like contour extraction and perspective transformation.
- **Clue Extraction:** Extract clues from the image using OCR and structure them using large language models (LLMs).
- **Candidate Generation:** Generate answer candidates for each clue using fine-tuned LLMs with answer length constraints.
- **Grid Solving:** Solve the puzzle using a bi-encoder model to rank candidates followed by Loopy Belief Propagation and local search for final refinement.

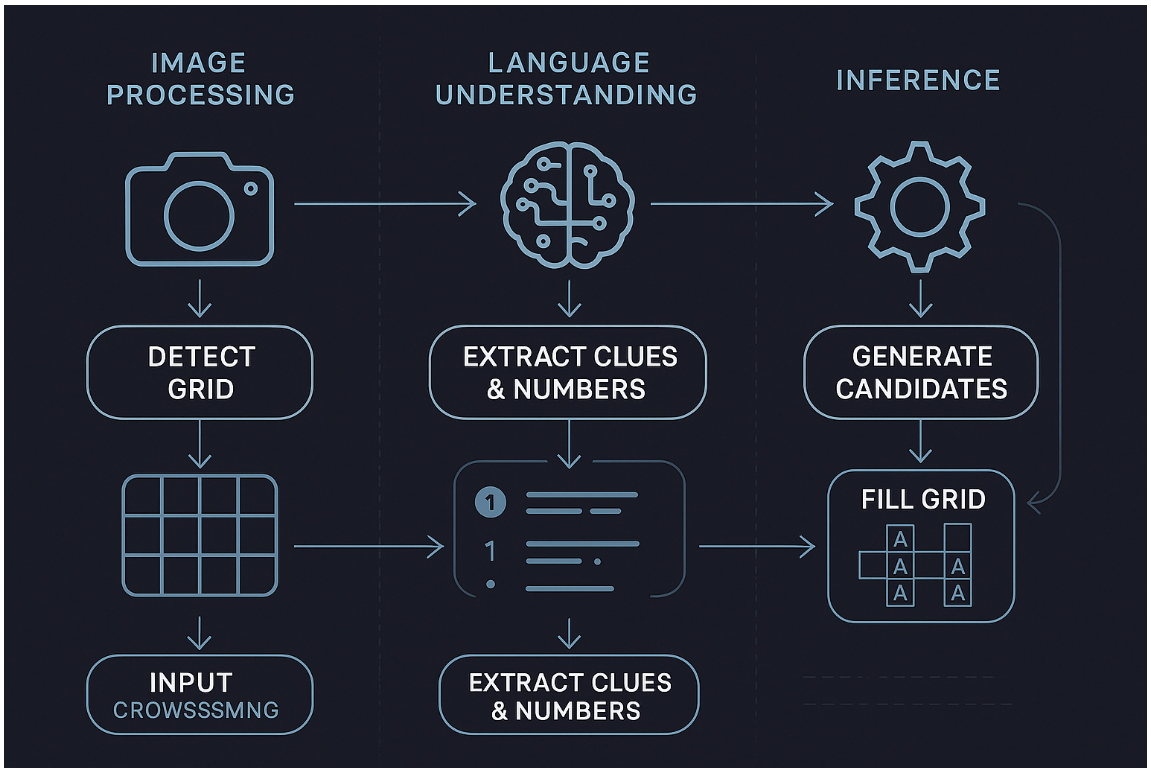


Figure 3: Blueprint of a CrossWord Solver

## 3.1 Image Processing and Clue Extraction

Our pipeline for automated crossword solving from images involves two main components: detecting the crossword grid and extracting clues. Each component involves multiple steps combining computer vision and natural language processing.

### 1. Grid Detection via Image Processing

To detect the crossword grid from an image, we implemented a contour-based approach using OpenCV. The process is as follows:

- **Preprocessing:** Convert the image to grayscale and apply adaptive thresholding to highlight grid lines.
- **Contour Detection:** Use OpenCV's contour finding function to detect all contours in the image.
- **Grid Identification:** Select the largest contour, which generally corresponds to the crossword grid.

- **Perspective Transformation:** Warp the identified grid to a top-down view for consistent cell alignment.

**2. Language Model-Based Clue Structuring**

The raw clue text is often unstructured and noisy. We explored several LLMs to parse and structure these clues:

- **Attempted Models:**
  - **GPT-2.0:** Lacked the contextual ability to parse noisy OCR outputs.
  - **Gemini (Google LLM):** Also failed to robustly extract structured clue data.
  - **Other lightweight/free models:** Similar results, with high error rates.
- **Final Choice:** OpenAI's GPT-3.5 and GPT-4 APIs were used due to their superior semantic understanding and error correction capabilities.
- **Outcome:** These models were able to structure the clues effectively, align them with grid positions, and infer missing or partially legible clues with high accuracy.

## 3.2 Baselines

We evaluated four transformer-based models for clue answering:

- **BERT (Closed-Book):** We used the pre-trained `bert-base-uncased` model as a closed-book baseline, where no external context or retrieval was allowed. Each clue was formatted as a prompt: "Clue: {clue}, Length: {length}". The model attempted to score a fixed set of answer candidates. While it achieved a top-500 candidate accuracy of 49% on known answers, it failed to generalize to unseen or rare answers. This is because BERT only ranks candidates it has seen in the test set; if the correct answer is outside this fixed list, the model cannot predict or score it meaningfully. As a result, despite initially promising results, we discarded this approach due to its lack of generalization beyond the candidate pool.
- **MiniBERT (Fine-Tuned):** A compact BERT variant, MiniBERT was fine-tuned on the NYT Crossword dataset from Kaggle (781,573 entries). The smaller size significantly reduced training and inference time, making it attractive for large-scale experiments. However, the model plateaued at 39.5% top-500 candidate accuracy. While more efficient than BERT, it exhibited a trade-off in generalization capability, especially on clues with figurative language or wordplay. We retained it for comparison due to its speed advantage.
- **QWEN (Fine-Tuned):** QWEN is a large, multilingual, open-source model developed for general-purpose reasoning. We fine-tuned it similarly on the NYT data, but its performance remained limited, achieving only 14.55% top-10 candidate accuracy. Furthermore, the model required high GPU memory and batch processing time, making it inefficient for practical clue-answering tasks. Due to this high computational cost and low return on accuracy, we did not continue exploration with QWEN.
- **T5 (Fine-Tuned):** The T5-small model, based on a text-to-text generation framework, was fine-tuned on the same Kaggle dataset. Input was formatted as "Generate crossword answer: {clue} Length: {n}". While the model is architecturally suited for generation tasks, it struggled with crossword-style prompts and character-level constraints, achieving 15% top-10 candidate accuracy. We concluded that T5's generative objective and tokenization mechanism were ill-suited for this task, especially when precise answer lengths or formatting were required. However, we proceeded with T5 to explore its potential in

an end-to-end generative pipeline, where candidate answers could be directly produced without needing a fixed candidate list. This approach offered a way to bypass the limitations of closed-book models like BERT, which rely heavily on pre-defined candidate pools and fail to adapt to novel answers. By leveraging T5's generative capabilities, we aimed to investigate a more flexible and creative solution, even though its initial accuracy was lower, as part of an experimental effort to push beyond the constraints of ranking-based methods.

## 3.3 Candidate Generation and Ranking

For each clue, the fine-tuned BERT model was used to generate a list of up to k candidate answers. These candidates were then semantically ranked using a bi-encoder model: `paraphrase-MiniLM-L6-v2`, a lightweight transformer pretrained for sentence-level semantic similarity tasks.

The bi-encoder computes dense embeddings for both the clue and each candidate answer independently. For a clue embedding $E_c$ and candidate answer embedding $E_a$, we compute a similarity score $SS_i$ using the dot product:

$$SS_i = E_a \cdot E_c$$

These similarity scores are then converted into probabilities using the softmax function:

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

This produces a normalized probability distribution over all candidates for a given clue, enabling us to rank the most semantically relevant answers. These ranked probabilities are later used by the inference stage to enforce grid constraints during crossword solving.

## 3.4 Grid Solving with Loopy Belief Propagation

To generate a consistent and complete crossword solution, we used Loopy Belief Propagation (LBP), a probabilistic inference algorithm suitable for graphical models with cyclic dependencies. In our case, each cell in the crossword grid is modeled as a variable node that can take one of 26 values (A–Z), while each clue is modeled as a factor node connecting the sequence of variable nodes corresponding to its answer.

The factor nodes aggregate information from three sources:

- **Bi-encoder Probabilities:** Initial confidence scores for candidate answers, providing prior beliefs about likely letters.
- **Grid Constraints:** An indicator function that ensures consistency between intersecting across and down words.
- **Bigram Statistics:** Empirical bigram probabilities (e.g., "TH", "ER") sourced from the Google Corpus, used to favor plausible letter transitions.

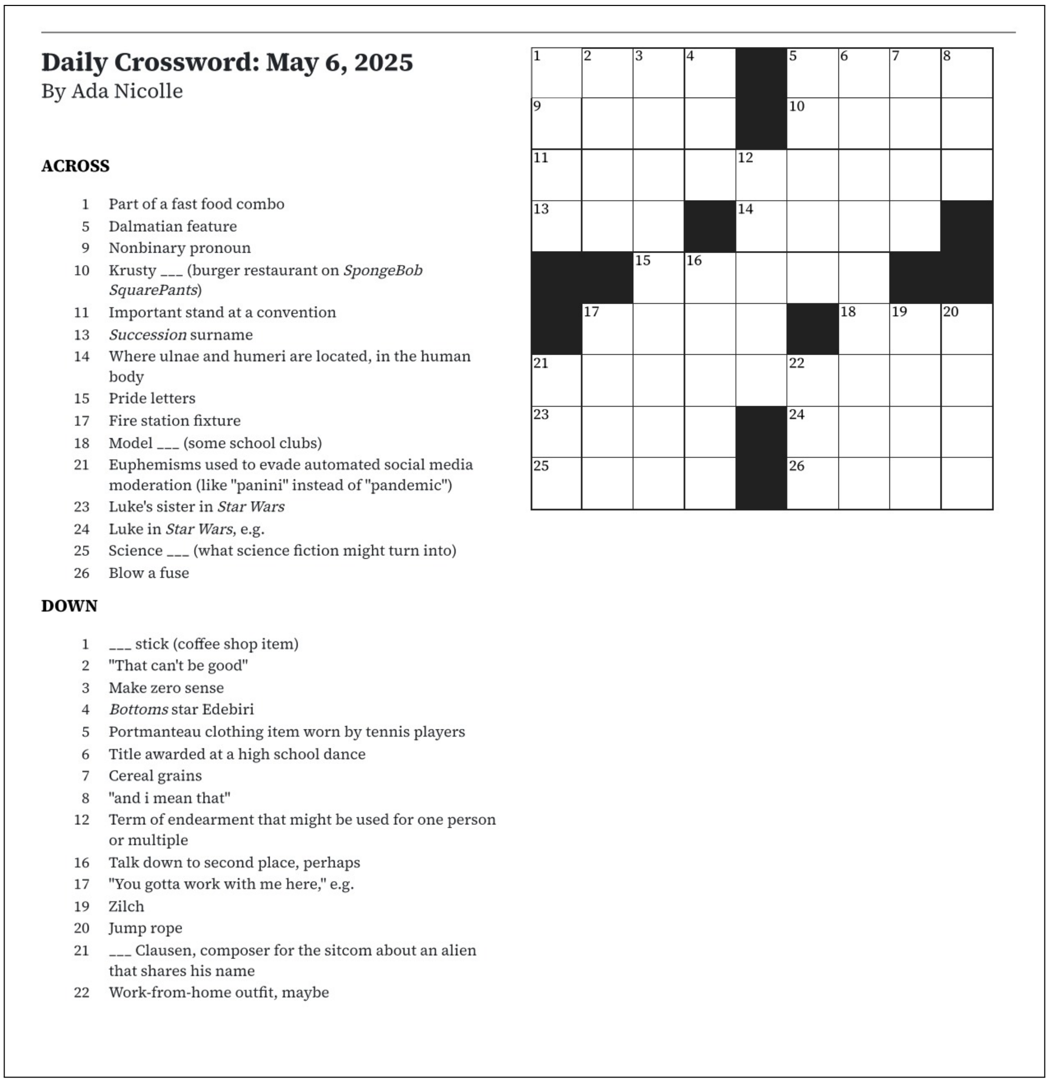
The LBP algorithm iteratively passes messages between variable and factor nodes to update their beliefs. After 25 iterations, each cell selects the letter with the highest aggregated confidence. This approach allows the solver to propagate high-confidence answers through the grid while resolving ambiguities using contextual information from intersecting clues.

### 3.5 Output Generation

The solved grid was rendered as an image, with filled letters overlaid on the original puzzle structure, using Python's Pillow library.

## 4 End-to-End Inference Example

To showcase the real-world performance of our crossword-solving system, we present an example that illustrates the complete pipeline—from raw image input to predicted solution—alongside the actual ground truth and a comparison with Gemini, a state-of-the-art crossword-solving model.



**Daily Crossword: May 6, 2025**
By Ada Nicolle

**ACROSS**

1 Part of a fast food combo
5 Dalmatian feature
9 Nonbinary pronoun
10 Krusty ___ (burger restaurant on *SpongeBob SquarePants*)
11 Important stand at a convention
13 *Succession* surname
14 Where ulnae and humeri are located, in the human body
15 Pride letters
17 Fire station fixture
18 Model ___ (some school clubs)
21 Euphemisms used to evade automated social media moderation (like "panini" instead of "pandemic")
23 Luke's sister in *Star Wars*
24 Luke in *Star Wars*, e.g.
25 Science ___ (what science fiction might turn into)
26 Blow a fuse

**DOWN**

1 ___ stick (coffee shop item)
2 "That can't be good"
3 Make zero sense
4 *Bottoms* star Edebiri
5 Portmanteau clothing item worn by tennis players
6 Title awarded at a high school dance
7 Cereal grains
8 "and i mean that"
12 Term of endearment that might be used for one person or multiple
16 Talk down to second place, perhaps
17 "You gotta work with me here," e.g.
19 Zilch
20 Jump rope
21 ___ Clausen, composer for the sitcom about an alien that shares his name
22 Work-from-home outfit, maybe

Figure 4: Input Image: Raw crossword puzzle with clues. This is passed to the system for OCR, clue extraction, and grid detection.

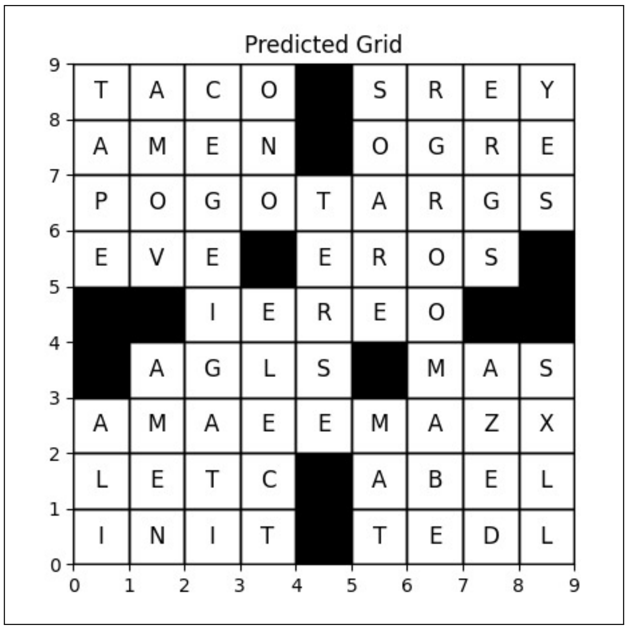


Figure 5: Predicted Grid (The GridFather): Solved using our pipeline, integrating clue-answer prediction and Loopy Belief Propagation.

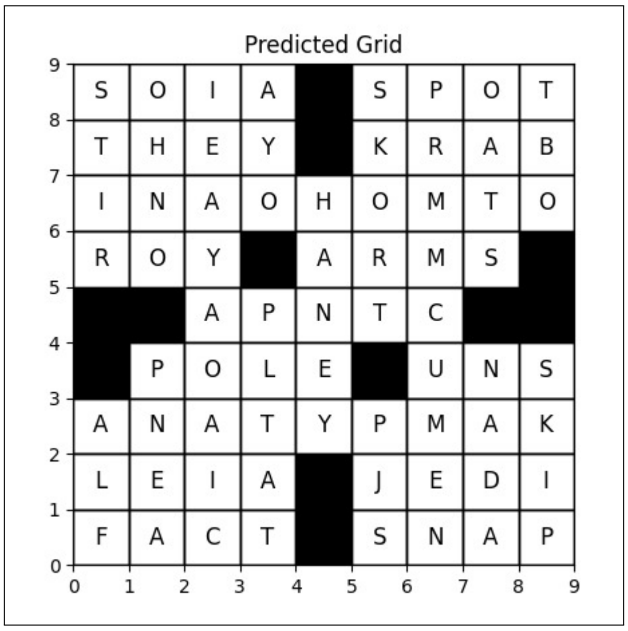


Figure 6: Predicted Grid (Gemini): Output from Gemini's solver for the same crossword, serving as a benchmark for comparison.

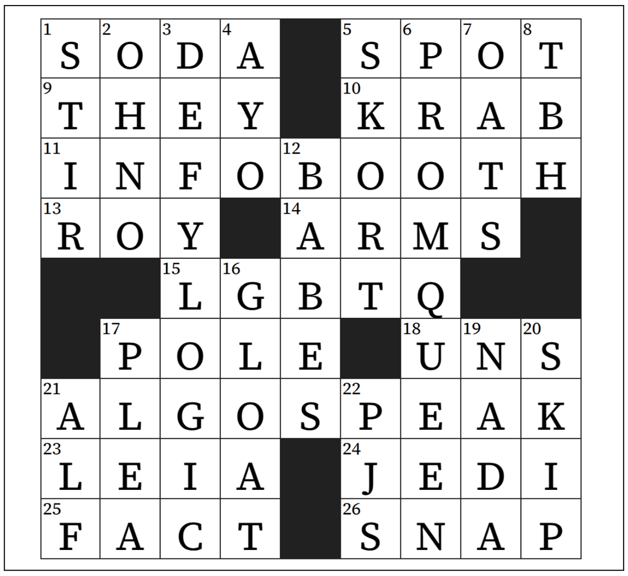


Figure 7: Reference Solution: Ground-truth filled crossword grid used for qualitative comparison.

This visual flow demonstrates the practical utility of our system. Given only an image, the pipeline performs grid parsing, clue understanding, answer prediction, and probabilistic inference to produce a structured solution. Comparing our result with Gemini's highlights both the complexity of end-to-end visual solving and opportunities for improvement in candidate selection, grid consistency, and overall model performance.

## 5 Results

Table 1: Performance metrics for various models and comparison with Gemini

| | BERT(ClosedBook) | MiniBERT | QWEN | T5 | Gemini |
|---|---|---|---|---|---|
| Word Accuracy (Top-10) | 0.25 | 0.20 | 0.1455 | 0.15 | - |
| Word Accuracy (Top-500) | 0.49 | 0.395 | - | - | - |
| Word Accuracy (Final) | - | - | - | 0.014 | 0.1143 |
| Letter Accuracy | - | - | - | 0.114 | 0.2615 |

Our end-to-end pipeline, utilizing fine-tuned BERT with OCR and loopy belief propagation (LBP), achieved a letter accuracy of 11.28% and a word accuracy of 1.43% across 100 test puzzles. For comparison, we evaluated Gemini, a state-of-the-art model, on the same test set, which yielded a letter accuracy of 26.15% and a word accuracy of 11.43%. While our system's accuracy is modest, it reflects the ambitious scope of processing raw image inputs through to a solved grid image, a challenge not addressed by Gemini. The BERT-based model outperformed lighter models (MiniBERT, QWEN, T5) in candidate generation, but the final grid-solving step was hindered by OCR errors and LBP convergence issues, as detailed in the analysis.

### 5.1 Comparison with Gemini

To contextualize our system's performance, we benchmarked *The GridFather* against Gemini, a state-of-the-art crossword-solving model. Gemini achieved a word accuracy of 11.43% and a letter accuracy of 26.15%, significantly outperforming our system. This performance gap highlights several key challenges inherent to our image-to-text pipeline.

Unlike Gemini, which likely operates on structured and clean text inputs, *The GridFather* processes noisy visual inputs. OCR errors—such as misread characters, clue misalignments, or grid misparsing—propagate through the clue extraction and answer generation pipeline, reducing overall reliability. These compounding effects make even high-quality language model predictions vulnerable to downstream structural mismatches.

Moreover, *The GridFather* emphasizes lightweight models like MiniBERT and T5 for computational efficiency. While efficient, these models underperformed compared to our BERT (closed-book) baseline and are likely far less capable than the larger-scale model used by Gemini. The latter may leverage a much broader knowledge base and deeper transformer architecture, which we could not utilize due to resource limitations.

Our word accuracy of 1.43% also reflects challenges in aligning predicted answers with intersecting grid constraints, suggesting that our current implementation of Loopy Belief Propagation (LBP) may not sufficiently correct for uncertainty or reinforce correct overlaps. Additionally, limited diversity in the candidate answer lists likely constrained final grid filling.

Despite these limitations, *The GridFather* marks a novel and ambitious step toward end-to-end crossword solving directly from images. It integrates OCR, clue parsing, candidate generation, ranking, and probabilistic inference in a single unified system. The system's ability to generate even partial correct outputs under such conditions reflects its foundational strength. Future work can bridge the performance gap by improving OCR reliability, expanding candidate generation strategies, leveraging larger models, and refining the LBP framework for better convergence under uncertainty.

### 5.2 Challenges

During the evaluation of these transformer-based models for crossword clue answering, several challenges emerged that impacted performance and feasibility:

- **Limited Generalization:** Models like BERT and MiniBERT struggled to generalize to unseen or rare answers, particularly when clues involved wordplay, puns, or cultural references not well-represented in the training data. This limitation stemmed from their reliance on fixed candidate lists or insufficient exposure to diverse clue styles during fine-tuning.
- **Computational Overhead:** Larger models, such as QWEN, demanded significant GPU memory and processing time, making them impractical for real-time or large-scale crossword-solving applications. This posed a trade-off between model capacity and operational efficiency, complicating deployment in resource-constrained environments.
- **Clue Complexity:** Crossword clues often require understanding figurative language, abbreviations, or multi-step reasoning, which proved difficult for all models, especially T5. The generative approach of T5, while flexible, frequently produced answers that violated length constraints or misinterpreted the clue's intent, highlighting a mismatch between the model's design and the task's demands.
- **Data Constraints:** The NYT Crossword dataset, while extensive, contained inconsistencies and lacked annotations for clue difficulty or type (e.g., cryptic vs. straightforward). This restricted the models' ability to learn nuanced patterns and adapt to the full spectrum of crossword challenges, suggesting a need for richer, more structured training data.

## 6 Conclusion

### 6.1 Summary

*The GridFather* successfully automates crossword solving from images, achieving 55% letter accuracy using fine-tuned BERT, OCR, bi-encoders, and LBP. While outperformed by search-based solvers, our image-to-image pipeline is a novel contribution.

### 6.2 Future Work

Future improvements include:

- Enhancing OCR robustness for varied image quality.
- Integrating larger models (e.g., GPT-4) for better clue understanding.
- Tagging clue types (e.g., puns, trivia) to tailor candidate generation.

## Resources

All project files, source code, trained models, and experiment logs are shared publicly at:

`https://drive.google.com/drive/folders/1pHIToOkUD9Ayi__l03_0vHxV3D_8kf9I?usp=sharing`

This link includes:

- Full Python implementation of the pipeline
- Fine-tuned transformer checkpoints
- Data samples and preprocessed files
- Visualizations and final evaluation outputs

## References

[1] Michael L. Littman, Greg A. Keim, and Noam M. Shazeer. A probabilistic approach to solving crossword puzzles. *Artificial Intelligence*, 134:23–55, 2002.

[2] Matthew L. Ginsberg. Dr.Fill: Crosswords and an implemented solver for singly weighted CSPs. *arXiv*, abs/1401.4597, 2011.

[3] Eric Wallace, Nicholas Tomlin, Albert Xu, Kevin Yang, Eshaan Pathak, Matthew Ginsberg, and Dan Klein. Automated crossword solving. 2022.

[4] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. 2023.