

Empirical Evaluation of Automated and Expert-Guided Prompt Engineering for Structured Data Extraction in Large Language Models

Aditya Minz¹, Rajat Bose¹, Rishav Gupta¹, Neha Gupta²

1 School of Engineering and Technology, Sushant University, Gurgaon, Haryana, India.

*2 Associate Professor, School of Engineering and Technology, Sushant University, Gurgaon, Haryana, India,
orchid id 0000-0002-1201-777X.*

Abstract

Large language models (LLMs) are increasingly used for structured data extraction, yet their performance is highly sensitive to the wording of the input prompt. This paper presents a controlled comparison of three prompting strategies for extracting structured JSON from noisy receipt and invoice text: "a minimal baseline prompt", "a carefully crafted human expert prompt", and "an automated prompt optimizer that iteratively refines a prompt over five iterations using failure analysis and LLM-driven rewriting". All three approaches are evaluated on a 20-sample dataset of synthetically noised receipts, using deterministic metrics including JSON validity, exact match, and field-level accuracy across seven schema fields. The baseline prompt achieved 12.9% field accuracy, the human expert prompt reached 90.4%, and the best automated optimizer iteration achieved 84.3% on the held-out test set. The human expert approach achieved the highest test-set field-level accuracy, outperforming the alternative by 6.1% in absolute terms. The results contribute to the growing understanding of whether automated prompt engineering can substitute for, or complement, manual prompt crafting in production extraction pipelines.

Keywords: *Prompt Engineering, Large Language Models, Automated Optimization, Structured Data Extraction, Natural Language Processing, Neural Networks, Blockchain, Intelligent Computing Systems*

1. Introduction

Large language models have demonstrated remarkable capabilities in structured information extraction, converting unstructured or semi-structured text into well-defined JSON objects that downstream systems can consume [1]. However, the quality of extraction is critically dependent on how the task is communicated to the model through its input prompt. This sensitivity has given rise to the discipline of prompt engineering — the systematic design and refinement of prompts to elicit desired model behavior [2, 3].

Manual prompt engineering, while effective, is labor-intensive. A human expert must understand the task domain, anticipate edge cases, encode extraction rules, and iterate through multiple prompt versions. Recent work on automated prompt optimization offers an alternative: algorithms that treat the prompt itself as an optimizable artifact, using the LLM to rewrite and improve its own instructions based on performance feedback [4, 5, 6, 7].

The foundation of these capabilities lies in artificial neural networks (ANNs), which have evolved significantly over the past decades. Kaur and Gupta [8] provide a comprehensive review of ANNs, tracing their development from simple perceptrons to the complex deep learning architectures that underpin modern LLMs. Their work highlights how the fundamental principles of neural computation — learning from data, generalizing to unseen inputs, and adapting through iterative optimization — are directly applicable to the prompt optimization paradigm investigated in this study.

This raises a natural research question: Can automated prompt optimization match or exceed the performance of a human expert prompt for structured data extraction? Despite growing interest in automated approaches such as DSPy [6], MIPRO [7], APE [4], and OPRO [5], controlled head-to-head comparisons against human-crafted prompts remain scarce, particularly for structured extraction tasks with deterministic evaluation.

In this paper, a controlled experiment is presented comparing three prompting conditions on a receipt/invoice extraction task: (1) A minimal baseline prompt with no task-specific guidance; (2) A human expert prompt incorporating established best practices; and (3) An automated optimizer that iteratively improves a prompt over five rounds, starting from the baseline.

The contributions of this paper are: (a) A controlled comparison framework with fully deterministic evaluation metrics (no LLM-as-judge), enabling exact reproducibility; (b) An analysis of what automated optimization discovers compared to human intuition, including prompt evolution trajectories and overfitting diagnostics; and (c) Practical cost-benefit analysis of automated vs. manual prompt engineering.

The remainder of this paper is organized as follows. Section 2 reviews related work on prompt engineering, automated optimization, neural network foundations, and intelligent computing. Section 3 details the methodology, including the task definition, dataset, evaluation metrics, and experimental setup. Section 4 presents results across all three conditions. Section 5 discusses implications, limitations, and directions for future work. Section 6 concludes.

2. Literature Review

2.1 Foundations of Neural Networks and Intelligent Computing

The theoretical underpinnings of modern large language models are rooted in decades of research on artificial neural networks. Kaur and Gupta [8] provide a foundational review of artificial neural networks, examining their architectures, learning paradigms, and applications across diverse domains. Their survey traces the evolution from early perceptron models to multi-layer networks capable of learning complex non-linear mappings, establishing the conceptual framework upon which today's transformer-based models are built.

Building upon this foundation, Kaur and Gupta [9] propose a constructive neural network framework that addresses key challenges in network design, particularly the problem of determining optimal network topology. Constructive approaches begin with minimal network structures and incrementally add neurons and connections based on learning performance, offering a principled alternative to the trial-and-error approach of manual architecture design. This concept of iterative, performance-driven construction bears a notable parallel to the automated prompt optimization pipeline presented in this paper, which similarly starts from a minimal configuration and iteratively adds complexity.

Further advancing constructive neural network methodologies, Kaur and Gupta [10] introduce a bipolar sigmoid algorithm specifically designed for constructive neural networks. The bipolar sigmoid activation function, with its output range of $[-1, 1]$, offers advantages in terms of convergence speed and the ability to represent both positive and negative correlations in data. Their experimental results demonstrate improved training efficiency compared to standard sigmoid-based approaches, highlighting the importance of activation function selection in neural network performance — a consideration analogous to the role of specific prompt elements in LLM performance.

The extended bipolar sigmoid algorithm proposed by Kaur and Gupta [11] further enhances the performance of constructive neural networks by incorporating additional optimization techniques that improve both convergence rate and generalization capability. Their work shows that systematic refinement of algorithmic components can yield measurable performance gains, a principle that directly informs the investigation in this paper of whether systematic, automated refinement of prompts can similarly improve LLM extraction accuracy.

2.2 Blockchain and Intelligent Systems

The intersection of intelligent computing with emerging technologies such as blockchain has opened new avenues for secure, decentralized applications. Dewan, Duhan, and Gupta [12] propose a secure electronic voting system based on mobile applications and blockchain technology, demonstrating how intelligent

systems can be combined with distributed ledger technology to address trust and security concerns in democratic processes. Their work illustrates the broader trend of applying computational intelligence to real-world systems requiring high integrity and transparency.

Dargan, Gupta, and Singh [13] present a blockchain-based energy management system model that leverages intelligent computing for efficient energy distribution and monitoring. By combining blockchain's immutable record-keeping with smart algorithms for energy optimization, their proposed model addresses critical challenges in modern energy grids, including fraud prevention, real-time monitoring, and decentralized decision-making. This interdisciplinary approach exemplifies the convergence of AI, blockchain, and domain-specific computing.

Dargan and Gupta [14] further explore the integration of blockchain for IoT (Internet of Things) communication, examining how blockchain technology can enhance the security, privacy, and reliability of IoT networks. Their work addresses the fundamental challenge of trust in distributed sensor networks and proposes architectural frameworks that combine cryptographic security with intelligent data processing. The concepts of automated verification and optimization discussed in their work parallel the automated prompt optimization techniques studied in this paper.

2.3 Prompt Engineering

Manual prompt engineering has emerged as a critical skill for effective LLM deployment. White et al. [3] catalog common prompt patterns including role assignment, output formatting constraints, and chain-of-thought reasoning. Wei et al. [2] demonstrate that chain-of-thought prompting substantially improves multi-step reasoning by instructing models to show intermediate steps. Wang et al. [15] extend this with self-consistency, sampling multiple reasoning paths and marginalizing over them. These techniques form the foundation of the human expert prompt used in this experiment.

Reynolds and McDonnell [16] investigate the relationship between prompt design and zero-shot task performance, demonstrating that carefully crafted prompts can unlock latent capabilities in pre-trained models without any fine-tuning. Their findings underscore the importance of prompt engineering as a lightweight alternative to model retraining. Liu et al. [17] present a comprehensive survey of pre-train, prompt, and predict paradigms, systematically categorizing prompt-based learning methods and their relationship to traditional fine-tuning approaches.

Kojima et al. [18] make the remarkable discovery that simply prepending "Let's think step by step" to a prompt dramatically improves reasoning performance, establishing that LLMs are inherently "zero-shot reasoners" when appropriately prompted. This finding has profound implications for prompt optimization, suggesting that small textual changes can produce outsized effects on model behavior.

2.4 Automated Prompt Optimization

Several lines of work automate the prompt engineering process. Zhou et al. [4] introduce Automatic Prompt Engineer (APE), which uses an LLM to generate and score candidate instructions, selecting the best-performing variant. Yang et al. [5] propose OPRO, framing prompt optimization as a meta-optimization problem where the LLM acts as its own optimizer. Khattab et al. [6] develop DSPy, a programming framework that compiles declarative LM programs into optimized prompts. Opsahl-Ong et al. [7] extend DSPy with MIPRO, jointly optimizing instructions and few-shot demonstrations for multi-stage pipelines. Fernando et al. [19] explore evolutionary approaches in PromptBreeder, where prompts self-referentially evolve over generations.

Pryzant et al. [20] introduce automatic prompt optimization that uses gradient-guided search in the text space, providing a bridge between continuous optimization methods and discrete prompt engineering. Their approach demonstrates that optimization principles from deep learning can be adapted for prompt refinement. Shin et al. [21] propose AutoPrompt, which automatically generates prompts by leveraging gradient-based search to identify optimal trigger tokens that elicit desired model behavior without manual engineering.

Deng et al. [22] present RLPrompt, which formulates prompt generation as a reinforcement learning problem, training a policy network to select optimal prompt tokens. This approach offers the advantage of considering long-term rewards and exploration-exploitation trade-offs in prompt design. Sun et al. [23] propose Black-Box Tuning, which optimizes continuous prompts for language models without requiring access to model gradients, making it applicable to API-only LLM access scenarios similar to the experimental setup presented in this paper.

2.5 Structured Data Extraction with LLMs

LLMs are increasingly applied to structured extraction tasks such as named entity recognition, relation extraction, and form parsing. Brown et al. [1] demonstrate that GPT-3 can perform few-shot extraction across diverse formats. Ye et al. [24] specifically study prompt engineering for extraction tasks, finding that task decomposition and explicit schema specification significantly improve accuracy. This paper contributes to this area by directly comparing manual and automated prompting strategies on a controlled extraction benchmark.

Li et al. [25] provide a comprehensive survey of deep learning approaches for named entity recognition (NER), covering distributed representations, network architectures, and decoding methods. Their systematic analysis of how neural models extract structured entities from unstructured text provides theoretical grounding for the LLM-based extraction task studied in this paper. Wei et al. [26] investigate the emergent abilities of large language models — capabilities that appear only at sufficient model scale —

demonstrating that structured reasoning and extraction competencies emerge as model parameters increase, a phenomenon directly relevant to the extraction performance observed with gpt-4o-mini.

Josifoski et al. [27] present GenIE, a generative approach to information extraction that frames the task as constrained text generation, specifically targeting knowledge graph construction from text. Their approach of constraining model output to valid structured formats is closely related to the schema-adherence evaluation metric used in this study. Wang et al. [28] propose ChatIE, a zero-shot information extraction framework that leverages multi-turn conversational interaction with ChatGPT to extract structured information, demonstrating that conversational prompting strategies can effectively guide LLMs toward producing schema-compliant structured outputs.

2.6 Evaluation of LLM Outputs

Evaluating structured LLM outputs presents unique challenges. Unlike free-text generation where human judgment or LLM-as-judge approaches are common, structured extraction allows deterministic evaluation through field-level comparison against ground truth. The evaluation framework employed in this paper scores JSON validity, exact match, and per-field accuracy, enabling precise quantification of differences without introducing evaluator variance.

Zheng et al. [29] propose MT-Bench and Chatbot Arena for evaluating LLM conversation quality, establishing benchmarking frameworks that complement the deterministic evaluation approach adopted in this study. Chang et al. [30] provide a comprehensive survey on evaluation methods for LLMs, categorizing approaches by task type and discussing the trade-offs between automated metrics, human evaluation, and LLM-based judging. Their taxonomy contextualizes the choice of deterministic metrics as the gold standard for structured extraction evaluation.

Liang et al. [31] present the HELM framework for holistic evaluation of language models, advocating for multi-metric assessment that captures different dimensions of model capability. This philosophy informs the multi-metric evaluation design in this paper, which combines JSON validity (syntactic correctness), exact match (holistic accuracy), and field-level accuracy (fine-grained performance analysis) to provide a comprehensive view of extraction quality.

3. Methodology

3.1 Task Definition

The task is to convert noisy receipt and invoice text into a structured JSON object conforming to a predefined schema. Input texts simulate realistic conditions including OCR errors, abbreviations, missing

delimiters, and formatting artifacts. The model must produce a valid JSON object with seven top-level fields: vendor_name, date, total, currency, line_items, tax, and payment_method.

This schema requires the model to perform multiple sub-tasks simultaneously: entity recognition (vendor name, payment method), date normalization, numerical extraction (totals, prices, tax), array construction (line items), and null handling (missing fields).

3.2 Dataset

The dataset consists of 20 manually constructed receipt and invoice samples. Each sample pairs a noisy input text with a verified ground-truth JSON object. The samples are designed to cover a range of realistic noise types, as summarized in Table 1.

Table 1: Noise types and their descriptions used in the synthetic receipt dataset.

Noise Type	Description	Example
OCR character errors	Common character confusions from scanning	"0" for "O", "1" for "l"
Abbreviations	Shortened field names and values	"qty", "amt", "pymt"
Missing delimiters	Absent or inconsistent separators	No column alignment
Extra whitespace	Irregular spacing and garbage characters	"~~~", "***", spaces
Reordered fields	Fields in non-standard order	Tax before subtotal
Missing fields	Nullable fields absent (~5 samples)	No tax or payment listed
Ambiguous dates	Formats could be MM/DD or DD/MM	"03/04/2025"
Mixed currency	Different currency indicators	"\$", "EUR", pound sign

The dataset is split into 10 training and 10 test samples using a fixed random shuffle (seed=42). The training set is used by the automated optimizer to evaluate and improve prompts; the test set is held out for final evaluation of all three conditions to measure generalization.

3.3 Evaluation Metrics

All evaluation is fully deterministic — no LLM-as-judge or human scoring is involved. This study employs four metrics:

JSON Validity (binary): Whether the model output can be parsed as a valid JSON object. The parser attempts direct parsing, markdown fence stripping, and regex extraction as fallbacks.

Exact Match (binary): Whether the parsed JSON, after normalization (lowercase strings, round floats to 2 decimal places, sort line items by description), is identical to the ground truth.

Field-Level Accuracy (per-field, 0–1): Each of the seven top-level fields is scored independently. Scalar fields receive 1.0 for exact match and 0.0 otherwise. The aggregate field accuracy is the mean across all seven field scores.

Token Usage and Cost: Total prompt and completion tokens consumed, converted to estimated USD cost using published API pricing.

3.4 Experimental Setup

All experiments use the fixed configuration summarized in Table 2.

Table 2: Fixed experimental configuration parameters.

Parameter	Value
Model	gpt-4o-mini
Temperature	0.0
Max tokens	1024
Random seed	42
Dataset size	20 samples
Train/test split	10 / 10

Temperature is set to 0.0 to ensure deterministic outputs, and a fixed seed is used for reproducibility. Three experimental conditions are compared:

Baseline: A minimal prompt that provides no task-specific guidance, schema information, or error correction instructions.

Human Expert: A single carefully crafted prompt incorporating role assignment, explicit schema specification, OCR error correction rules, date normalization conventions, and output format constraints.

Automated Optimizer: An iterative optimization pipeline that starts from the baseline prompt and performs 5 improvement iterations, each guided by failure analysis on the training set. A 500word limit prevents prompt bloat.

3.5 Automated Optimization Pipeline

The automated optimizer follows an iterative improvement loop. At each iteration, it evaluates the current prompt on the training set, analyzes failures (identifying which fields were wrong and why), and asks an optimizer LLM to generate an improved prompt. The optimizer model receives the current prompt, training-

set performance metrics, detailed failure analysis (per-sample wrong fields with expected vs. actual values), and the target JSON schema.

Key design choices include: (a) Bloat control — a 500-word limit is enforced; if the optimizer produces a prompt exceeding this limit, it is asked to compress while preserving key instructions. (b) Overfitting control — the optimizer only sees training-set performance; test-set evaluation is performed post hoc to measure generalization. (c) Placeholder preservation — the prompt must contain the `{input_text}` placeholder where receipt text is inserted at inference time.

4. Results

4.1 Main Comparison

Table 3 presents the primary comparison across all three experimental conditions on the held-out test set.

Table 3: Main comparison of the three prompting strategies on the test set.

Method	JSON Validity	Exact Match	Field Accuracy (Mean)	Field Accuracy (Std)
Baseline	100.0%	0.0%	12.9%	0.1383
Human Expert	100.0%	45.0%	90.4%	0.1043
Optimizer (Best)	100.0%	30.0%	84.3%	0.1421

The human expert prompt achieved the highest field-level accuracy on the test set (90.4%), outperforming the automated optimizer (84.3%) by 6.1 percentage points. The baseline prompt achieved only 12.9%, confirming that structured prompting is essential for reliable extraction. All three methods achieved 100% JSON validity, indicating that the model consistently produces parseable JSON regardless of prompt quality.

4.2 Optimizer Trajectory

Table 4 shows how the optimizer’s performance evolved across iterations on both training and test sets.

Table 4: Optimizer trajectory across iterations. * denotes the iteration selected as best by training-set performance

Iteration	Train Accuracy	Test Accuracy	Prompt Words
0 (Baseline)	11.4%	14.3%	8
1	20.0%	12.9%	17
2 *	82.4%	84.3%	119

3	82.4%	87.1%	119
4	81.0%	82.9%	135
5	78.1%	84.3%	135

The most dramatic improvement occurs between iterations 1 and 2, where field accuracy jumps from 20.0% to 82.4% on training data. This corresponds to the optimizer discovering the importance of schema specification and structured output formatting. Subsequent iterations yield marginal improvements, with some degradation observed in iterations 4 and 5, suggesting potential instability in the optimization process at later stages.

4.3 Field-Level Analysis

Table 5 presents the per-field accuracy breakdown, revealing distinct performance profiles for each method.

Table 5 : Field-level accuracy breakdown across all methods on the test set.

Field	Baseline	Human Expert	Optimizer (Best)
currency	0.0%	80.0%	90.0%
date	15.0%	85.0%	90.0%
line_items	0.0%	97.5%	90.0%
payment_method	40.0%	75.0%	60.0%
tax	35.0%	100.0%	100.0%
total	0.0%	95.0%	100.0%
vendor_name	0.0%	100.0%	60.0%

The human expert prompt excels at vendor_name (100% vs. 60%) and line_items (97.5% vs. 90.0%), fields requiring OCR error correction and complex structure parsing respectively. The optimizer achieves superior performance on currency (90% vs. 80%), date (90% vs. 85%), and total (100% vs. 95%), which are fields with more standardized formats amenable to automated pattern discovery. Both methods achieve 100% accuracy on tax extraction.

4.4 Statistical Significance

To assess whether the difference between the human expert and optimizer is statistically significant, multiple statistical tests are applied to paired per-sample field accuracy scores on the test set, as presented in Table 6.

Table 6: Statistical significance tests comparing human expert and optimizer performance.

Test	Value
Wilcoxon W statistic	3.500
Wilcoxon p-value	0.375
McNemar chi-squared	0.000
McNemar p-value	1.000
Bootstrap 95% CI (diff)	[-0.1714, 0.0143]
Test	Value
Cohen's d	-0.463 (small effect)

The Wilcoxon signed-rank test ($p = 0.375$) and McNemar's test ($p = 1.000$) do not reach statistical significance at the conventional threshold ($\alpha = 0.05$), primarily due to the small sample size ($n = 10$). The bootstrap 95% confidence interval for the difference in field accuracy $([-0.17, 0.01])$ marginally includes zero, indicating that the advantage of the human expert is suggestive but not statistically conclusive at this sample size. Cohen's d of -0.463 indicates a small-to-medium effect size in favor of the human expert, warranting further investigation with larger datasets.

4.5 Cost Analysis

Table 7 summarizes the computational cost of each approach.

Table 7: Cost analysis comparing token usage and estimated costs across methods

Method	Extraction Tokens	Optimization Tokens	Total Cost (USD)
Baseline	5,041	0	\$0.0021
Human Expert	11,570	0	\$0.0031
Optimizer	56,292	24,210	\$0.0224

The automated optimizer consumed approximately 7.2x the total cost of the human expert prompt (\$0.0224 vs. \$0.0031), driven by repeated evaluation across 5 iterations plus the optimization calls themselves. The average cost per iteration was approximately \$0.0037. However, this purely computational cost does not account for the human labor required to craft the expert prompt, which may be substantially more expensive in production settings where prompt engineering expertise commands premium rates.

5. Discussion

5.1 Interpretation of Results

The human expert approach achieved the highest overall field-level accuracy on the held-out test set (90.4% vs. 84.3%). This suggests that human domain knowledge and prompt engineering expertise still provide a meaningful edge, particularly for tasks where the expert can anticipate common failure modes (OCR errors, date formats, abbreviations) and encode correction rules directly into the prompt. The automated optimizer, while effective, may be constrained by the quality of its failure analysis and the optimizer LLM's ability to translate error observations into targeted prompt improvements.

Notably, both the human expert and automated optimizer dramatically outperform the minimal baseline (12.9%), confirming that prompt engineering — whether manual or automated — is essential for reliable structured extraction. This finding aligns with the broader neural network literature, where Kaur and Gupta [8, 9] demonstrate that network architecture and training methodology significantly impact model performance, regardless of the underlying computational substrate.

The constructive neural network approaches proposed by Kaur and Gupta [10, 11] offer an illuminating parallel: just as their algorithms iteratively build optimal network topologies from minimal starting points using bipolar sigmoid activation functions, the automated optimizer presented in this paper iteratively constructs optimal prompts from minimal instructions using failure-driven refinement. The success of both approaches suggests that iterative, performance-driven construction is a broadly applicable optimization paradigm across different levels of the AI stack.

5.2 Overfitting Analysis

The best optimizer iteration (iteration 2) achieved 82.4% field accuracy on the training set and 84.3% on the test set, yielding a train-test gap of -1.9 percentage points (test slightly outperformed train). Across all iterations, the maximum train-test gap was 7.1 percentage points (iteration 1), and no iteration exhibited a gap exceeding the 10% overfitting threshold commonly used in machine learning practice.

This relatively small gap suggests that the optimizer generalized well from the training set to unseen test samples. The prompt improvements appear to capture general extraction principles rather than sample-specific corrections, an encouraging finding for the practical applicability of automated prompt optimization.

5.3 Prompt Evolution

The optimizer's prompt evolved substantially across iterations, growing from 8 words to 135 words. Key elements that emerged through optimization include: (a) Schema specification — the optimizer

independently learned to include the target JSON schema, mirroring the human expert's approach; (b) Error correction rules — OCR error patterns were identified from failure analysis and encoded in the prompt; (c) Output format constraints — instructions to return only raw JSON were added after observing markdown fence artifacts in early iterations; and (d) Null handling — explicit instructions for nullable fields emerged after failures on samples with missing tax or payment information.

5.4 Failure Mode Comparison

Baseline: Fails broadly across all extraction dimensions. Without schema guidance, the model invents field names, omits nullable fields entirely, and often wraps output in markdown fences. Date normalization and OCR correction are inconsistent.

Human Expert: Failures are concentrated in edge cases: complex line item parsing with unusual formatting, ambiguous date resolution when context clues are insufficient, and occasional currency inference errors when no explicit currency symbol is present in the input text.

Automated Optimizer: Failure patterns are iteration-dependent. Early iterations resemble baseline failures; later iterations converge toward human-expert-like failures. The optimizer may also introduce novel failure modes if it over-corrects for specific training-set errors, such as hardcoding vendor name corrections that do not generalize.

5.5 Implications for Intelligent Computing Systems

The findings of this study have broader implications for the design of intelligent computing systems. The principles of iterative optimization demonstrated in the prompt engineering pipeline directly resonate with the constructive neural network methodologies of Kaur and Gupta [9, 10, 11], where performance-driven construction leads to efficient, task-appropriate architectures. In both paradigms, the system starts minimal and grows in complexity guided by empirical performance feedback, a strategy that appears effective across fundamentally different optimization domains.

Furthermore, the integration of such automated optimization techniques with blockchain-based systems [12, 13, 14] could enable trusted, auditable AI pipelines where prompt versions, optimization histories, and performance metrics are immutably recorded on a distributed ledger. As Dewan et al. [12] demonstrate with secure electronic voting and Dargan et al. [13, 14] illustrate with energy management and IoT communication, blockchain provides the transparency and immutability infrastructure needed for high-stakes AI applications. This is particularly relevant for domains such as healthcare, finance, and governance where AI-driven extraction must be transparent, reproducible, and auditable.

5.6 Limitations

This study has several important limitations that should be considered when interpreting the results:

(1) Small dataset — with only 20 samples (10 train / 10 test), statistical power is limited, and results may not generalize to larger, more diverse receipt corpora. (2) Single domain — all samples are receipts and invoices; performance patterns may differ for other structured extraction tasks such as medical records, legal documents, or scientific data. (3) Single model — all experiments use gpt-4o-mini; results may vary with other models (GPT-4, Claude, Llama, etc.) that have different instruction-following characteristics and extraction capabilities.

(4) Human prompt authorship — the human expert prompt was authored by the researchers, not crowdsourced from multiple prompt engineers; a different expert might produce a more or less effective prompt. (5) No adversarial testing — the dataset does not include adversarial examples designed to exploit specific prompt weaknesses. (6) Fixed optimizer budget — the optimizer runs for exactly 5 iterations; additional iterations might yield further improvements or might exacerbate overfitting. (7) No few-shot examples — neither the human nor the optimizer prompt includes few-shot demonstrations, which could substantially alter the performance landscape.

6. Conclusion

This paper presented a controlled comparison of three prompting strategies — a minimal baseline, a human expert prompt, and an automated prompt optimizer — for structured JSON extraction from noisy receipt and invoice text, evaluated on a held-out test set of 10 samples using fully deterministic metrics.

The comparative results reveal a clear performance hierarchy. The minimal baseline, with no task-specific guidance, achieved only 12.9% field-level accuracy, confirming that unstructured prompting is insufficient for reliable extraction. The automated optimizer, starting from this same minimal baseline and iteratively refining over five iterations, reached 84.3% field-level accuracy and 30.0% exact match — a 71.4 percentage point improvement over the baseline achieved entirely without human intervention. The human expert prompt achieved the highest performance at 90.4% field-level accuracy and 45.0% exact match, outperforming the automated optimizer by 6.1 percentage points in field accuracy and 15.0 percentage points in exact match.

A field-level comparison further highlights the complementary strengths of each approach. The human expert prompt excelled at fields requiring domain-specific reasoning, such as `vendor_name` (100% vs. 60%) and `line_items` (97.5% vs. 90.0%), where anticipation of OCR errors and complex structural parsing proved critical. Conversely, the automated optimizer achieved superior performance on fields with standardized

formats — currency (90% vs. 80%), date (90% vs. 85%), and total (100% vs. 95%) — demonstrating that automated pattern discovery is effective for well-structured data elements. Both methods achieved identical performance on tax extraction (100%).

Despite the observed performance gap, statistical tests (Wilcoxon $p = 0.375$, McNemar $p = 1.000$) did not reach significance at $\alpha = 0.05$, and the bootstrap 95% confidence interval $([-0.17, 0.01])$ marginally includes zero, indicating that the difference, while consistent, is not statistically conclusive at this sample size. From a cost perspective, the automated optimizer incurred approximately $7.2\times$ the computational cost of the human expert (\$0.0224 vs. \$0.0031), though this does not account for the human labor investment required to craft the expert prompt.

The parallels between the prompt optimization methodology presented in this paper and the constructive neural network frameworks of Kaur and Gupta [9, 10, 11] suggest a unifying principle: iterative, performance-driven construction from minimal starting points is an effective optimization strategy across different levels of the AI stack, from neural architectures to prompt engineering. Combined with the transparency and auditability afforded by blockchain technologies [12, 13, 14], these approaches point toward a future of more reliable, trustworthy, and automatically optimized intelligent computing systems.

Future work should investigate: (a) scaling to larger and more diverse datasets to improve statistical power; (b) extending to multiple extraction domains including medical records, legal documents, and financial statements; (c) hybrid human-optimizer pipelines where the optimizer refines a human-authored starting prompt; (d) multi-model evaluation to test prompt transferability across different LLM architectures; (e) longer optimization horizons with early stopping criteria to balance improvement against overfitting; and (f) integration with blockchain-based audit trails for production deployment of optimized extraction pipelines.

7. References

- [1] Brown, T. B., Mann, B., Ryder, N., et al. "Language Models are Few-Shot Learners." *Advances in Neural Information Processing Systems* 33 (2020): 1877–1901.
- [2] Wei, J., Wang, X., Schuurmans, D., et al. "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models." *Advances in Neural Information Processing Systems* 35 (2022): 24824–24837.
- [3] White, J., Fu, Q., Hays, S., et al. "A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT." *arXiv preprint arXiv:2302.11382* (2023).
- [4] Zhou, Y., Muresanu, A. I., Han, Z., et al. "Large Language Models Are Human-Level Prompt Engineers." *International Conference on Learning Representations* (2023).
- [5] Yang, C., Wang, X., Lu, Y., et al. "Large Language Models as Optimizers." *International Conference on Learning Representations* (2024).

- [6] Khattab, O., Singhvi, A., Maheshwari, P., et al. "DSPy: Compiling Declarative Language Model Calls into SelfImproving Pipelines." International Conference on Learning Representations (2024).
- [7] Opsahl-Ong, K., Ryan, M. J., Hardy, J., and Khattab, O. "Optimizing Instructions and Demonstrations for MultiStage Language Model Programs." arXiv preprint arXiv:2406.11695 (2024).
- [8] Kaur, J., and Gupta, N. "Artificial Neural Network: A Review." International Journal of Technical Research & Science (n.d.): 1–4.
- [9] Kaur, J., and Gupta, N. "Constructive Neural Network: A Framework." International Journal of Engineering and Advanced Technology 9(2) (2019). <https://doi.org/10.35940/ijeat.b3304.129219>
- [10] Kaur, J., and Gupta, N. "Bipolar Sigmoid Algorithm for Designing Constructive Neural Network." International Journal on Emerging Technologies 11(2) (2020): 991–996.
- [11] Kaur, J., and Gupta, N. "Extended Bipolar Sigmoid Algorithm for Enhancing Performance of Constructive Neural Network." International Journal on Emerging Technologies 11(2) (2020): 1034–1038.
- [12] Dewan, S., Duhan, L., and Gupta, N. "Secure Electronic Voting System Based on Mobile App and Blockchain." Recent Advances in Computer Science and Communications 14(9) (2021). <https://doi.org/10.2174/2666255813999200821161058>
- [13] Dargan, J., Gupta, N., and Singh, L. "Blockchain-Based Energy Management System: A Proposed Model." In 2021 International Conference on Technological Advancements and Innovations (ICTAI), pp. 510–514. IEEE (2021). <https://doi.org/10.1109/ICTAI53825.2021.9673233>
- [14] Dargan, J., and Gupta, N. "Integration of Blockchain for IoT Communication." In Adaptive Power Quality for Power Management Using Smart Technologies, pp. 319–343. CRC Press (2023).
- [15] Wang, X., Wei, J., Schuurmans, D., et al. "Self-Consistency Improves Chain of Thought Reasoning in Language Models." International Conference on Learning Representations (2023).
- [16] Reynolds, L., and McDonell, K. "Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm." Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems (2021).
- [17] Liu, P., Yuan, W., Fu, J., et al. "Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing." ACM Computing Surveys 55(9) (2023): 1–35.
- [18] Kojima, T., Gu, S. S., Reid, M., et al. "Large Language Models Are Zero-Shot Reasoners." Advances in Neural Information Processing Systems 35 (2022): 22199–22213.
- [19] Fernando, C., Banarse, D., Michalewski, H., Osindero, S., and Rocktaschel, T. "PromptBreeder: Self-Referential Self-Improvement via Prompt Evolution." arXiv preprint arXiv:2309.16797 (2023).
- [20] Pryzant, R., Iter, D., Li, J., et al. "Automatic Prompt Optimization with Gradient Descent and Beam Search." Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (2023).
- [21] Shin, T., Razeghi, Y., Logan IV, R. L., Wallace, E., and Singh, S. "AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts." Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (2020): 4222–4235.
- [22] Deng, M., Wang, J., Hsieh, C. P., et al. "RLPrompt: Optimizing Discrete Text Prompts with Reinforcement Learning." Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (2022): 3369–3391.
- [23] Sun, T., Shao, Y., Qian, H., et al. "Black-Box Tuning for Language-Model-as-a-Service." Proceedings of the 39th International Conference on Machine Learning (2022): 20841–20855.
- [24] Ye, S., Hwang, D., Kim, S., et al. "Prompt Engineering a Prompt Engineer." ACL 2024 Workshop on Natural Language Reasoning and Structured Explanations (2024).
- [25] Li, J., Sun, A., Han, J., and Li, C. "A Survey on Deep Learning for Named Entity Recognition." IEEE Transactions on Knowledge and Data Engineering 34(1) (2022): 50–70.
- [26] Wei, J., Tay, Y., Bommasani, R., et al. "Emergent Abilities of Large Language Models." Transactions on Machine Learning Research (2022).

- [27] Josifoski, M., De Cao, N., Peez, M., Petroni, F., and Riedel, S. "GenIE: Generative Information Extraction." Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics (2022): 4626–4643.
- [28] Wang, S., Zhao, Z., Ouyang, X., Wang, Q., and Cai, D. "ChatIE: Zero-Shot Information Extraction via Chatting with ChatGPT." arXiv preprint arXiv:2302.10205 (2023).
- [29] Zheng, L., Chiang, W. L., Sheng, Y., et al. "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena." Advances in Neural Information Processing Systems 36 (2023).
- [30] Chang, Y., Wang, X., Wang, J., et al. "A Survey on Evaluation of Large Language Models." ACM Transactions on Intelligent Systems and Technology 15(3) (2024): 1–45.
- [31] Liang, P., Bommasani, R., Lee, T., et al. "Holistic Evaluation of Language Models." Transactions on Machine Learning Research (2023).