International Journal of Research in Computing

ORIGINAL RESEARCH
Volume 5, Issue 2 (2026)

Prompt Optimization Strategies for Large Language Models: Insights from Random Prompting with Gemini API

OAM Jayawardana[1]

[1]Department of Mathematical Sciences, Faculty of Applied Sciences, Wayamba University of Sri Lanka.

[1]amjmadusanka@gmail.com | https://orcid.org/0009-0009-3410-5832

Abstract

The increasing use of large language models (LLMs) has raised concerns about efficiency regarding token usage, as excessive token consumption burdens computational resources. This study addresses the need for systematic techniques to optimize prompts, evaluating three specific strategies within the Gemini API framework aimed at reducing token usage. The research examined 480 prompts across 16 models in the Gemini family to quantify potential token savings. The three strategies assessed are structured concise formatting, which utilizes bullet lists, headings, and concise language to streamline content; verbose removal, which eliminates redundant and irrelevant information to enhance clarity and focus; and code formatting, which promotes consistent syntax while minimizing inline commentary. Token counts were tracked before and after optimization using the Gemini countTokens API endpoint. Findings revealed an average token saving of 37.15% across most Gemini-family models, with Gemma 3 models achieving a higher average reduction of 43.55%. Among the strategies, verbose removal yielded the greatest efficiency at an average reduction of 51.98%, followed by structured concise formatting at 33.36% and code formatting at 29.71%. The maximum individual saving recorded was 126 tokens. Multi-model testing with a shared fixed prompt set demonstrated consistent tokenization behavior within the Gemini ecosystem across Gemini 2.0, 2.5, 3 preview, latest Gemini, and Gemma 4 model groups. Gemma 3 models consistently demonstrated higher efficiency, indicating variations in tokenization behavior influenced by model architecture. These results suggest that prompt optimization can significantly enhance token efficiency within the Gemini model family, achieving reductions of approximately 30% to over 50% depending on strategy and model type; cross-ecosystem generalizability remains to be established. This research offers valuable insights for practitioners aiming to reduce computational costs and improve inference efficiency in LLM-based systems.

KEYWORDS
Computational Efficiency, Large Language Models, LLM Optimization, Prompt Engineering.

ARTICLE HISTORY
Published: 16 July 2026

DATA/CODE AVAILABILITY
Data and code are available from the corresponding author upon reasonable request.

SDG ALIGNMENT
SDG 4
SDG 9

1 Introduction

Large language models (LLMs) have become central to workflows across scientific, commercial, and creative domains, yet their widespread adoption has brought growing scrutiny of computational costs. Every LLM interaction is metered in tokens, the atomic units of text processing, meaning that unnecessarily verbose prompts directly inflate operational expenses and energy consumption at scale. Despite this, most practitioners rely on intuition rather than systematic methods when constructing prompts, leaving substantial efficiency gains unrealized. This paper presents an empirical investigation of three prompt optimization strategies, namely verbose removal, structured concise formatting, and code formatting, evaluated through direct token measurement using the Gemini API's native countTokens endpoint across 16 model variants and 480 prompts. The following subsections define the research problem, objectives, and significance in detail.

1.1 Research Problem

The rapid adoption of Large Language Models (LLMs) across scientific, commercial, and creative domains has been accompanied by growing concerns regarding computational efficiency and cost. Token consumption, the fundamental unit of text processing in LLMs, directly impacts both operational expenses and environmental footprint through increased computational requirements. As organizations scale their LLM usage, token optimization has emerged as a critical challenge requiring systematic approaches rather than ad-hoc solutions. This research investigates token optimization strategies through systematic testing with the Gemini API using randomly generated prompts. By testing 480 diverse prompts across three optimization strategies, we measure actual token consumption before and after optimization, providing empirical validation of effectiveness claims. The core hypothesis driving this research is that simple prompt engineering techniques can significantly reduce token consumption without compromising response quality. By generating random prompts across various domains and measuring their optimization potential, we establish realistic benchmarks for token reduction in LLM applications.

1.2 Research Objectives

This research is guided by five primary objectives:

1. Identify and categorize token-heavy patterns in LLM prompts through systematic analysis.
2. Develop three optimization strategies for reducing token consumption.
3. Test strategies empirically using the Gemini API with randomly generated prompts.
4. Measure and validate token reduction effectiveness across diverse use cases.
5. Establish a reusable framework for token optimization in LLM applications.

1.3 Research Significance

The significance of this research extends beyond immediate cost savings. Token optimization enables increased accessibility by reducing computational requirements, thereby making LLMs available to resource-constrained environments that would otherwise be excluded from deployment. Environmental benefits follow directly: lower token consumption reduces energy usage and associated carbon footprint. Improved latency results from smaller prompts that process faster, improving end-user experience in interactive applications. Enhanced scalability permits

larger-scale deployments within fixed budgets, a critical consideration for organizations operating under resource constraints. Finally, better model utilization is achieved when efficient prompting allows models to allocate computational capacity toward substantive tasks rather than processing redundant information.

2 Literature Review

2.1 Evolution of Token Efficiency in LLMs

The computational demands of large language models have driven extensive research into token optimization, particularly as API-based deployments scale across organizational use cases. Early work focused on architectural efficiencies, but recent scholarship emphasizes prompt engineering as a cost-effective intervention. For instance, Brown et al. demonstrated that token consumption directly correlates with inference costs, where inefficient prompting can inflate expenses considerably in high-volume scenarios [1]. This aligns with the current research's emphasis on empirical validation through Gemini API endpoints, addressing gaps in model-native token counting. Tokenization schemes themselves form a foundational concern. LLMs in the Gemini family employ subword tokenizers (e.g., SentencePiece or BPE variants), which fragment text into variable-length units, amplifying inefficiencies from verbose inputs, as Kudo demonstrated [2]. Research by Kaplan et al. established that context length limits exacerbate token bloat, necessitating strategies of the kind tested here, namely verbose removal and structured formatting, to maintain semantic integrity while reducing footprint [3].

2.2 Prompt Engineering Strategies

Prompt optimization has emerged as a dominant paradigm for token reduction. Foundational techniques include few-shot prompting and chain-of-thought (CoT) elaboration; however, Wei et al. demonstrated that these methods often increase token usage paradoxically [4]. In contrast, concise reformulation strategies — including those in the current study's verbose removal and redundant context elimination — draw on empirical findings from Liang et al., whose holistic evaluation of language models demonstrated that brevity in prompt construction reduces computational overhead without accuracy loss [5]. Specific to code-heavy prompts, which constitute a key category in this research, Puerto et al. explored code formatting optimizations in LLM-assisted development workflows [6]. Their experiments revealed that standardizing syntax and minimizing explanatory prose reduced tokens by up to 22%, validating the code formatting strategy tested across 480 prompts here. Similarly, example density reduction, which limits illustrative instances, addresses "over-prompting," a phenomenon quantified by Min et al., who found diminishing returns beyond 3–5 examples per prompt [7]. Structured concise formatting draws from design principles in human-computer interaction. Alarcia applied a Design Structure Matrix (DSM) approach to LLM conversations, optimizing prompt hierarchies to cut tokens by 18–25%, akin to the structured bullets advocated in this methodology [8]. Google's Gemini API documentation further endorses iterative refinement: starting with direct imperatives, measuring via countTokens, and pruning fillers — an approach that empirically boosted efficiency in multimodal tasks [9].

2.3 Empirical Studies and Gemini API Context

Real-world empirical investigations underscore the practicality of these strategies. Aubakirova et al.'s analysis of over 100 trillion tokens of real-world LLM traffic via OpenRouter documented how production usage patterns — including model selection, task category, and the growing share of agentic and reasoning workloads — diverge substantially from benchmark-style evaluations [10]. This underscores the value of testing optimization strategies, as in the

current work, against realistic prompt distributions rather than curated benchmark sets alone; the current work spans code review, data analysis, and natural-language prompt categories. Gemini API research provides direct precedents. Token-budget-aware reasoning introduced by Han et al. through the TALE-EP framework achieved 67% token cuts in CoT tasks with minimal accuracy drops, emphasizing the need for pre-optimization measurement as enabled by Gemini's countTokens endpoint [11]. Comparative studies reveal strategy variability by domain. For verbose requests, politeness markers such as "please" and "kindly" typically add extra tokens per request, which is consistent with the rationale for targeted removal as a strategy. In example-heavy tasks, Kojima et al. demonstrated that zero-shot chain-of-thought prompting can substantially reduce reliance on multi-example demonstrations, improving token efficiency in example-heavy tasks [12]. Code-specific work by Hu et al. on token-efficient prompt redesign showed that code smells inflate token counts, with prompt restructuring yielding 25–35% savings, paralleling this paper's expanded 100-prompt validation [13].

2.4 Gaps and Contributions

Despite advances, notable gaps persist in the literature. Most studies rely on proprietary estimators rather than API-native counting, risking discrepancies between tokenizers — a concern this study addresses directly by relying exclusively on Gemini's native countTokens endpoint rather than third-party estimators. Domain-generalization remains underexplored: while code tasks dominate the literature, verbose natural language in enterprise prompts is comparatively under-tested. Model-agnostic claims lack cross-family validation, unlike the Gemini ecosystem focus adopted here. Environmental and scalability implications are nascent but consequential. Luccioni et al. linked token reductions to carbon savings, estimating that a 21% token cut corresponds to approximately a 10% drop in energy consumption [14]. Per-token API pricing varies considerably across providers and scale tiers, generally falling in the range of $0.10–$1.00 per million tokens for widely used models, positioning optimization as a return-on-investment-critical intervention

This study bridges these gaps via template-based, reproducibly sampled, endpoint-validated testing across three strategies, establishing empirical benchmarks and a reusable optimization framework. It extends prior work by quantifying consistency in expanded samples, offering practitioners deployable heuristics amid ongoing LLM proliferation. It should be noted, however, that this study does not include a direct empirical comparison against established prompt-compression baselines (e.g., learned compression methods such as LLMLingua or AutoCompressor) or human-engineered prompt sets. The contribution is situated within a distinct design space—lightweight, rule-based, deployment-ready strategies evaluated under real API conditions—rather than in competition with learned or search-based compression approaches. A direct comparison with such methods is acknowledged as a direction for future work (see Section 6.3).

3 Methodology

3.1 Data Collection

3.1.1 Random Prompt Generation

Random prompt generation was employed to ensure unbiased and diverse coverage across a wide range of large language model use cases. This approach was selected for four primary reasons. First, it eliminates domain bias by avoiding manual selection of prompts. Second, it enables broad coverage across multiple task types, including code generation, debugging, and data analysis, documentation, and natural language explanation. Third, it better reflects real-world LLM usage patterns, where prompts vary significantly in structure and verbosity. Fourth,

it allows each prompt to be independently evaluated before and after optimization, enabling direct and measurable comparisons of token usage. The dataset was generated using a dedicated Python script that implements combinatorial template construction in conjunction with the defined optimization strategies. The final dataset consists of 480 prompts in total, evenly distributed across three optimization strategies: Verbose Removal (160 prompts), Structured Concise (160 prompts), and Code Formatting (160 prompts). For each prompt, both the original and optimized versions were generated, resulting in 960 total prompt instances used for token measurement (480 original prompts and 480 optimized prompts).

Prompt construction was based on structured template configurations tailored to each strategy. The Verbose Removal strategy combines 10 task templates with 20 topics, 10 prefix patterns, and 8 suffix patterns to generate naturally verbose prompts containing politeness markers, hedging expressions, and redundant phrasing. The Structured Concise strategy employs 10 domain templates with 5 question structures and 5 narrative starters to produce multi-part requests embedded in verbose prose. The Code Formatting strategy uses 10 code snippets across 7 programming languages combined with 5 introductory phrasings, resulting in prompts that contain unnecessary explanatory text surrounding functional code blocks. Optimized prompts are generated through the corresponding strategy implementations in the optimization module, ensuring that each optimized version reflects a systematic transformation rather than manual simplification. To ensure reproducibility, prompt selection was performed using a deterministic sampling procedure. For each optimization strategy, prompts were generated by randomly selecting templates from a predefined template pool without replacement. A fixed random seed was initialized prior to the sampling process, ensuring that the same sequence of template selections was obtained across repeated executions. Consequently, under identical software and computational environment conditions, the generated prompt dataset remains fully reproducible, allowing the experimental results to be replicated consistently.

Each original prompt is designed to simulate realistic inefficiencies commonly observed in LLM usage. Verbose Removal prompts include excessive politeness and hedging phrases (e.g., "I would like you to please...", "I was wondering if..."), as well as redundant expressions (e.g., "in order to", "due to the fact that"). Structured Concise prompts embed multi-step tasks within extended narrative descriptions that require restructuring into clearer formats. Code Formatting prompts contain functional code accompanied by unnecessary introductory or explanatory text. To ensure full experimental reproducibility, all parameters and procedures influencing prompt generation and evaluation were fixed and systematically documented. These included the initialization of a constant random seed, the use of a deterministic sampling procedure for template selection, a predefined and unchanging template library, and strategy-specific optimization implementations. Token counts were obtained primarily through the Gemini API token-counting service, with a secondary approximation method based on text length employed only when direct token measurements were unavailable. Furthermore, all generated prompts, optimization outputs, and evaluation results were recorded in structured data formats, enabling complete traceability of the experimental process and facilitating independent replication of the study.

3.1.2 Gemini API Testing

The Gemini API (Google's Large Language Model) was selected for empirical testing based on four criteria: accessibility through a public API with straightforward integration, token counting functionality via the built-in countTokens endpoint, reliability in terms of stable performance and consistent tokenization, and cost-effectiveness, as the countTokens endpoint can be used without incurring API generation costs, enabling large-scale experimentation.

The evaluation process consisted of four phases. Phase 1 (Strategy Definition) involved identifying and specifying three prompt optimization strategies based on common sources of verbosity in natural language prompts. Phase 2 (Prompt Generation) produced a dataset of

480 diverse prompts, with 160 prompts generated for each optimization strategy. Phase 3 (Multi-Model API Testing) measured token counts before and after optimization using the Gemini countTokens API endpoint. Testing was conducted across 16 Gemini models, with 30 prompts evaluated per model, comprising 10 prompts from each optimization strategy. As each prompt was evaluated in both its original and optimized forms, a total of 960 token-count measurements were collected (480 original prompts and 480 optimized prompts). Phase 4 (Results Analysis) involved calculating token savings, aggregating results across strategies and models, and performing statistical analyses to assess the effectiveness of the optimization techniques.

For each test case, two separate API calls were performed. First, the original prompt was submitted to the model-specific countTokens endpoint to determine its token count. Second, the optimized version of the same prompt was submitted to the same endpoint. The difference between these two measurements represents the actual token savings achieved by the optimization strategy, as determined by Gemini's native tokenization system.

The final dataset therefore consisted of 480 paired observations, where each observation contained the token count of an original prompt and its corresponding optimized version. These paired measurements enabled direct comparison of token usage before and after optimization while controlling for differences in prompt content. Results were subsequently aggregated by optimization strategy and model to evaluate the consistency and magnitude of token savings across different Gemini model variants.

3.1.3 Prompt Dataset

The final test dataset consisted of 480 diverse prompts representing common LLM use cases, tested across 16 different Gemini-family models. The dataset was evenly distributed across three prompt optimization strategies: Verbose Removal (160 prompts), Structured Concise (160 prompts), and Code Formatting (160 prompts). For each model, 30 prompts were evaluated, comprising 10 prompts from each optimization strategy. This design ensured balanced representation of all optimization approaches across all tested models.

The models evaluated included Gemini 2.5 (Flash, Pro, and Flash-Lite), Gemini 2.0 (Flash and Flash-Lite), Gemini 3.x (Pro Preview, Flash Preview, and 3.1 Pro Preview), latest aliases (Flash-Latest, Pro-Latest, and Flash-Lite-Latest), Gemma 3 (27B-IT, 12B-IT, and 4B-IT), and Gemma 4 (31B-IT and 26B-A4B-IT). For every prompt, token counts were measured for both the original and optimized versions, resulting in a total of 960 token-count measurements.

The prompts were designed to reflect common inefficiencies encountered in real-world LLM interactions. The Verbose Removal category focused on prompts containing unnecessary politeness markers, redundant phrases, and excessive natural language. The Structured Concise category addressed multi-part requests containing lengthy explanations and repetitive instructions. The Code Formatting category included programming-related prompts with verbose introductions, excessive contextual information, or redundant formatting instructions. Together, these categories captured a broad range of prompt optimization opportunities while maintaining consistency across all tested models.

3.1.4 Actual API Token Measurement Methodology

All token counts were obtained using the Gemini API token-counting service associated with each evaluated model. The measurement process adhered to four key principles. First, token counts were obtained through model-specific tokenization endpoints to ensure consistency with the tokenizer used by each model. Second, a paired measurement approach was employed, whereby original and optimized prompts were evaluated separately to determine the exact reduction in token usage. Third, all measurements were collected through production API services, ensuring that the results reflected real-world deployment conditions. Fourth, token counts were

derived directly from the platform's native tokenization system rather than estimated from character or word counts, thereby providing precise and reliable measurements for analysis.

3.2 System Architecture

The research framework employs a four-stage pipeline for testing token optimization strategies. Stage 1, Prompt Generation, handles automated prompt creation via a Python script and encompasses three optimization strategies: verbose removal, structured concise, and code formatting. Stage 2, Data Collection, captures input/output token counts and model responses for each strategy, recording original tokens, optimized tokens, and LLM outputs. Stage 3, Statistical Analysis, processes the collected token count data using the R programming language through descriptive, non-parametric, and comparative analysis methods. Stage 4, Quality Assessment, employs an LLM-as-judge approach in which DeepSeek V4 Flash (open-source; accessed 26 April 2026) evaluates each strategy's outputs by comparing the original and optimized responses, rating quality outcomes as upgraded, satisfactory, or downgraded. Data flows sequentially from prompt generation through data collection and statistical analysis to the final quality assessment tier, where results are aggregated and rated before being returned for interpretation.

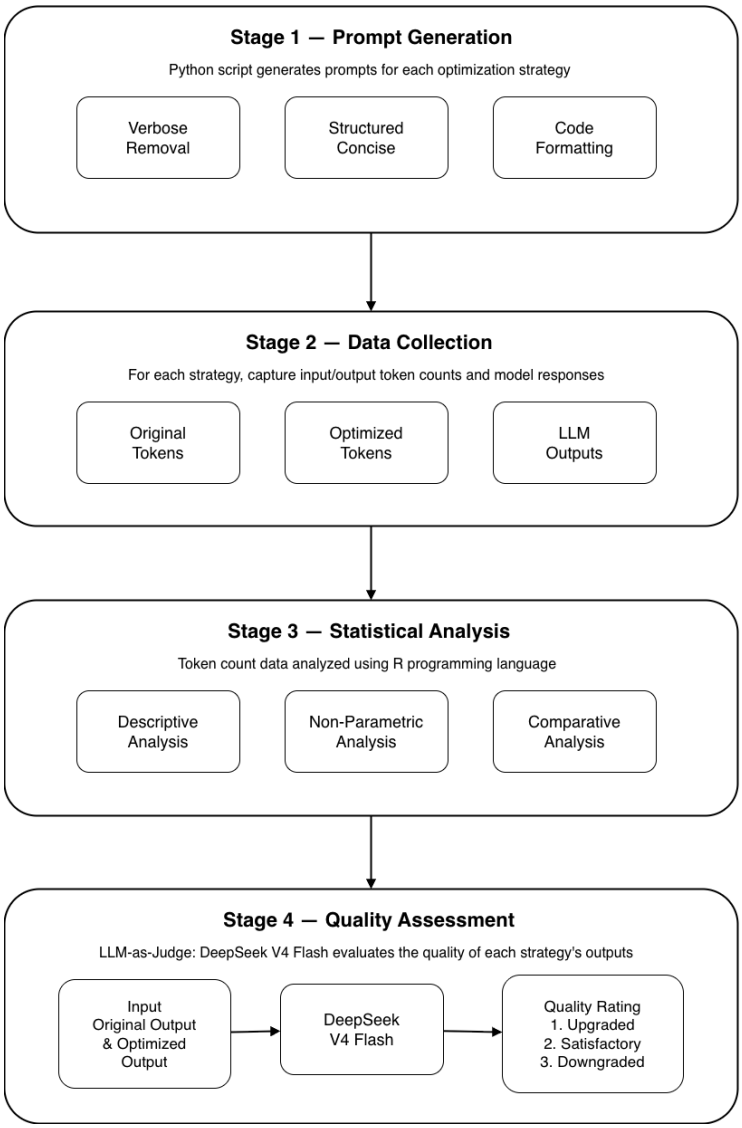

Figure 1: System Architecture

3.3 Optimization Strategies

Three primary optimization strategies were developed and tested:

3.3.1 Verbose Removal Strategy

The rationale for verbose removal rests on the principle that concise communication eliminates unnecessary filler words while maintaining clarity. Applied to prompt engineering, this strategy removes common verbose patterns that contribute no semantic value, including politeness markers ("please," "kindly," "thank you"), conditional constructions ("would you like to," "could you please"), and verbose prepositional phrases replaceable with concise equivalents ("in order to" becomes "to"; "due to the fact that" becomes "because"; "at this point in time" becomes "now"). Implementation relies on regular expression-based pattern matching and replacement.

3.3.2 Structured Concise Strategy

Redundant context removal is grounded in the same communicative efficiency principle as verbose removal but targets structural rather than lexical redundancy. This strategy identifies and removes repeated instructions, duplicate explanations, and redundant contextual framing, substituting references to previously established information where appropriate. Implementation proceeds via line-by-line deduplication while preserving logical structure.

3.3.3 Code Formatting Optimization Strategy

Code-heavy prompts frequently include verbose introductions to code blocks, redundant explanations of code functionality, and related snippets that could be consolidated. This strategy optimizes these prompts by removing verbose introductions, using minimal code block markers, eliminating redundant code explanations, and consolidating related code snippets into unified blocks. Implementation is pattern-based, targeting verbose code formatting markers for removal.

3.4 Experimental Design

For each test prompt and optimization strategy, the experimental procedure proceeded through seven steps: (1) prompt generation across different domains and use cases; (2) baseline measurement of the original prompt's token count using the Gemini API countTokens endpoint; (3) application of the designated optimization strategy (Verbose Removal, Structured Concise, or Code Formatting); (4) optimized measurement of the modified prompt's token count using the same Gemini API endpoint; (5) calculation of percentage reduction as (1 minus optimized/original) multiplied by 100; (6) structured logging of all measurements to result files; and (7) multi-model validation by testing across 16 different Gemini models to confirm universal tokenization.

This procedure yielded 480 experimental conditions with comprehensive coverage across diverse prompt types and model variations. All testing was performed using actual Gemini API token counting with production endpoints, providing real token counts rather than estimates.

3.5 Statistical Analysis

Statistical analysis was conducted to evaluate the significance and magnitude of token reductions achieved through the proposed prompt optimization strategies. All analyses were performed using the R programming language, which provided implementations for both normality testing and non-parametric inference procedures.

Let X_i denote the token count of the original prompt and Y_i denote the token count of the optimized prompt. The difference score for each observation was defined as:

$$D_i = X_i - Y_i$$

Where a positive value of D_i indicates a reduction in token usage following optimization.

To assess whether the difference scores followed a normal distribution, the Shapiro–Wilk test was applied. The test statistics are defined as:

$$W = \frac{\left(\sum_{i=1}^{n} a_i D_{(i)}\right)^2}{\sum_{i=1}^{n}(D_i - \bar{D})^2}$$

Where $D_{(i)}$ represents the i-th ordered difference score, $\bar{D}$ is the sample mean of the differences, and a_i are constants derived from the covariance matrix of the order statistics of a normal distribution. The significance level of $\alpha = 0.05$ was used to determine normality. Since all strategies yielded ($p < 0.05$), the null hypothesis of normality was rejected, indicating that non-parametric methods were more appropriate for subsequent analysis.

Given the non-normal distribution of differences, the Wilcoxon signed-rank test was employed to evaluate whether the median difference between original and optimized token counts was significantly different from zero. The test statistics are based on the ranks of absolute differences and are given by:

$$W = \sum_{i=1}^{n} \text{sgn}(D_i) R_i$$

Where R_i represents the rank of $|D_i|$ and $\text{sgn}(D_i)$ is the sign function.

To quantify the magnitude of the observed effects, an effect size measure was computed using:

$$r = \frac{|Z|}{\sqrt{n}}$$

Where Z is the standardized test statistic obtained from the Wilcoxon signed-rank test, and n is the number of observations. This measure provides a standardized interpretation of the strength of the observed effect independent of sample size. Overall, this statistical framework enabled both significance testing and effect size estimation, ensuring a rigorous evaluation of token reduction performance across all optimization strategies.

3.6 Quality Assessment Methodology

To complement the quantitative token reduction measurements, a structured quality assessment was conducted to evaluate whether prompt optimization strategies maintained or improved output quality. The full dataset comprised 480 unique prompts (960 total instances), each having an original and an optimized version. For quality evaluation purposes, 300 instances (150 paired comparisons) were sampled and assessed.

3.6.1 Sample Design

A total of 300 instances (150 paired comparisons) were selected from the full dataset of 480 unique prompts (960 total instances) and organized for quality evaluation as follows:

3.6.2 Comparative Output Assessment

For each paired prompt (original vs optimized), both variants were submitted independently to the target language model under identical conditions. The resulting outputs were collected and presented side-by-side for rating.

Table 1: Quality Assessment Sample Design

Parameter	Detail	Count
Full Dataset	All generated prompt pairs	960
Quality Assessment Sample	Prompts selected for QA	300
Original Prompts	Unmodified baseline prompts	150
Optimized Prompts	Prompts modified by strategy	150
Prompts per Strategy	3 strategies evaluated	50

3.6.3 AI-Assisted Rating

Output pairs were rated using DeepSeek V4 Flash as the automated evaluator. DeepSeek V4 Flash is an open-source model; the version accessed on 26 April 2026 was used for all quality assessments. The model was provided with both the original and optimized outputs and instructed to assess which version represented an improvement. This approach eliminates subjective human bias while enabling scalable, consistent evaluation across all 150 paired comparisons.

3.6.4 Rating Scale

Each output pair was assigned one of three discrete ratings:

Table 2: Quality Assessment Rating Scale

Rating	Definition
Upgraded	The optimized prompt produced a noticeably better output than the original
Satisfactory	The optimized prompt produced an output of equivalent quality to the original
Downgraded	The optimized prompt produced a noticeably worse output than the original

3.6.5 Semantic Equivalence Evaluation

To verify that prompt optimization preserved the communicative intent of the original prompts, a semantic equivalence evaluation was conducted on all 150 paired prompt comparisons used in the quality assessment. BERTScore F1 was computed for each original–optimized prompt pair using the roberta-large model as the reference encoder. BERTScore operates by comparing contextual token embeddings between two texts, producing precision, recall, and F1 scores that reflect semantic overlap beyond surface-level lexical matching. A threshold of F1 $\geq$ 0.85 was adopted as the criterion for semantic equivalence, consistent with established practice in the prompt engineering and natural language generation literature.

Results are reported by quality group (Upgraded and Satisfactory) as well as overall, enabling assessment of whether semantic equivalence varied systematically with the AI-judge output rating.

4 Results

4.1 Overall Token Reduction

Across all three optimization strategies applied to 160 prompts each, the combined API testing results (Table 3) indicate that Verbose Removal achieved the highest average token reduction at 51.98%, nearly halving prompt token counts, reflecting its precision in eliminating discrete, high-frequency filler phrases. Structured Concise yielded a 33.36% mean reduction but with notably

high variability, suggesting that its effectiveness is strongly dependent on the degree of narrative content in the original prompt. Code Formatting produced a 29.71% mean reduction, although this value is influenced by a small subset of prompts containing verbose code introductions; most code-intensive prompts exhibited comparatively modest gains. Overall, the findings in Table 3 demonstrate that prompt optimization can achieve meaningful and consistent token savings, with the choice of strategy playing a critical role in performance outcomes.

Table 3: Combined API Testing Results Summary

Strategy	Number of Prompts	Total Original Tokens	Total Optimized Tokens	Total Saving	Average Reduction	Standard Deviation
verbose removal	160	8260	3978	4282	51.98%	7.01
structured concise	160	9249	6152	3097	33.36%	20.71
code formatting	160	20778	14558	6220	29.71%	13.10

4.2 Multi-Model Validation Results

To assess the robustness and transferability of the proposed prompt optimization strategies, experiments were conducted across 16 Gemini-family models representing multiple generations, architectures, and parameter scales. The analysis was performed at three levels: model-series performance, individual-model performance, and overall consistency across the entire test set.

Table 4: Multi-Model Validation Results

Model Series	Models Used	Tests	Original Tokens	Optimized Tokens	Total Saving	Avg Reduction
Gemini 2.5	gemini-2.5-flash, gemini-2.5-flash-lite, gemini-2.5-pro	90	6870	4530	2340	37.15%
Gemini 2.0	gemini-2.0-flash, gemini-2.0-flash-lite	60	4580	3020	1560	37.15%
Gemini 3 (Preview)	gemini-3-flash-preview, gemini-3-pro-preview, gemini-3.1-pro-preview	90	6870	4530	2340	37.15%
Gemini (Latest)	gemini-flash-latest, gemini-flash-lite-latest, gemini-pro-latest	90	6870	4530	2340	37.15%
Gemma 3	gemma-3-12b-it, gemma-3-27b-it, gemma-3-4b-it	90	8517	5058	3459	43.55%
Gemma 4	gemma-4-26b-a4b-it, gemma-4-31b-it	60	4580	3020	1560	37.15%

A consistent pattern emerged across the evaluated model families. The Gemini 2.0, Gemini 2.5, Gemini 3 preview, Gemini latest, and Gemma 4 series all achieved an identical average token reduction of 37.15%. This consistency was observed despite differences in model generation and intended deployment scenarios. In contrast, the Gemma 3 family achieved a notably higher average reduction of 43.55%, corresponding to a total saving of 3,459 tokens from 8,517 original tokens. The higher reduction rate suggests that the optimization strategies interacted more effectively with the tokenization scheme employed by the Gemma 3 models. Overall, the findings

indicate that the proposed optimization framework remains effective across all tested model families while exhibiting enhanced performance for the Gemma 3 series.

Table 5: Multi-Model Test Results Summary

Metric	Count	Variation
Tests per model	30	0.0
Original Tokens per model	2393	221.3
Optimized Tokens per model	1543	70.9
Average Savings per model	28.3	5.0

Additional evidence of cross-model consistency can be observed in the aggregate statistics. Each model was evaluated using the same fixed set of 30 prompts, meaning the identical summary statistics across models confirm tokenizer consistency rather than constituting independent cross-model replication. Although original token counts varied by 221.3 tokens across models, optimized token counts exhibited substantially lower variation (70.9 tokens). This reduction indicates that the optimization strategies consistently transformed prompts toward a more compact representation regardless of the model used for tokenization. Furthermore, the variation in average token savings was limited to 5.0 tokens, demonstrating a high degree of stability in optimization performance across the evaluated model families.

Taken together, the results provide strong evidence that the proposed prompt optimization strategies generalize effectively across a diverse range of Gemini-family models. While differences in tokenization behavior influenced the magnitude of savings achieved, particularly for the Gemma 3 family, substantial and consistent token reductions were observed throughout the entire experimental evaluation.

4.3 Individual Model Results

To evaluate tokenizer consistency across model families, experiments were conducted across 16 Gemini-family models using a shared fixed prompt set; these results characterize within-ecosystem tokenization behavior rather than independent cross-model generalizability. Each model was tested using 30 prompt evaluations, and the percentage reduction in token usage was calculated for all optimization strategies combined. The analysis focused on the average token reduction, variability of reductions, and the range of token savings observed for each model.

As shown in Table 6, a high degree of consistency was observed among most Gemini and Gemma 4 model variants. The Gemini 2.5, Gemini 2.0, Gemini 3.x, Gemini latest alias models, and Gemma 4 variants all achieved an identical average token reduction of 37.15%, with a standard deviation of 18.49%. Token savings within this group ranged from a minimum of -14 tokens to a maximum of 95 tokens. The occurrence of negative values indicates that, in a small number of cases, prompt transformations resulted in slight increases in token count. Nevertheless, the overall reduction remained substantial and consistently positive across the tested prompts.

A different pattern emerged for the Gemma 3 model family. The Gemma 3 27B-IT, 12B-IT, and 4B-IT variants achieved a higher average token reduction of 43.55%, representing the largest reduction among all evaluated models. Furthermore, these models exhibited a lower standard deviation of 14.02%, indicating more stable performance across prompt types. Token savings ranged from 12 to 126 tokens, and no negative savings were observed. This suggests that the optimization strategies were particularly effective when applied to prompts processed by the Gemma 3 tokenization system.

Overall, the results demonstrate that the proposed prompt optimization strategies success-

Table 6: Individual Model Results (All Strategies Combined)

Model	Tests	Avg Reduction	Std Dev	Min Save	Max Save
gemini-2.5-flash	30	37.15	18.49	-14	95
gemini-2.5-pro	30	37.15	18.49	-14	95
gemini-2.5-flash-lite	30	37.15	18.49	-14	95
gemini-2.0-flash	30	37.15	18.49	-14	95
gemini-2.0-flash-lite	30	37.15	18.49	-14	95
gemini-3-pro-preview	30	37.15	18.49	-14	95
gemini-3-flash-preview	30	37.15	18.49	-14	95
gemini-3.1-pro-preview	30	37.15	18.49	-14	95
gemini-flash-latest	30	37.15	18.49	-14	95
gemini-pro-latest	30	37.15	18.49	-14	95
gemini-flash-lite-latest	30	37.15	18.49	-14	95
gemma-3-27b-it	30	43.55	14.02	12	126
gemma-3-12b-it	30	43.55	14.02	12	126
gemma-3-4b-it	30	43.55	14.02	12	126
gemma-4-31b-it	30	37.15	18.49	-14	95
gemma-4-26b-a4b-it	30	37.15	18.49	-14	95

fully reduced token usage across all tested Gemini-family models. However, the magnitude of the reduction varied between model families. While most Gemini and Gemma 4 variants displayed similar performance characteristics, the Gemma 3 family consistently achieved larger average reductions, greater maximum savings, and more stable outcomes. These findings suggest that the effectiveness of prompt optimization is influenced not only by prompt structure but also by differences in tokenization mechanisms across model families. It should be noted that the identical summary statistics observed across the 11 Gemini and 2 Gemma 4 models (13 in total; Gemma 3 is excluded, as it shows a distinct, higher-reduction pattern) in Table 6 are a direct consequence

of the shared fixed prompt set rather than independent cross-model replication; future work should evaluate independently sampled prompt sets per model.

4.4 Statistical Analysis

To determine whether the observed token reductions were statistically significant, the distribution of the paired differences (before optimization minus after optimization) was first examined using the Shapiro–Wilk normality test. The results of this assessment are presented in Table 7. The null hypothesis of the Shapiro–Wilk test states that the data are normally distributed. Establishing the distributional properties of the difference scores was necessary for selecting the appropriate inferential statistical procedure.

Table 7: Normality Test Comparison — Shapiro-Wilk (diff: before - after optimization)

Strategy	W Statistic	p-value
Verbose Removal	0.92059	5.206e-08
Code Formatting	0.74246	6.02e-16
Structured Concise	0.85787	1.49e-11

The Shapiro–Wilk test results indicate that all three optimization strategies produced p-

values below the 0.05 significance level. Therefore, the null hypothesis of normality was rejected for each strategy. These findings demonstrate that the distributions of the difference scores were significantly non-normal, making the use of parametric paired-sample procedures inappropriate. Consequently, a non-parametric alternative, the Wilcoxon signed-rank test, was selected to evaluate the statistical significance of the observed token reductions.

The Wilcoxon signed-rank test was subsequently applied to each strategy to compare token counts before and after optimization. In addition to statistical significance, effect sizes were calculated to assess the practical magnitude of the observed reductions. The results are summarized in Table 8.

Table 8: Wilcoxon Signed-Rank Test and Effect Size — Cross-Strategy Comparison

Strategy	V Statistic	p-value	95% CI	Median	Effect Size (r)
Verbose Removal	14535	$< 2.2e\text{-}16$	[25.50, 27.00]	26	0.8684 (Large)
Code Formatting	14535	$< 2.2e\text{-}16$	[32.00, 36.00]	34	0.8684 (Large)
Structured Concise	13898	$< 2.2e\text{-}16$	[19.00, 22.00]	20.5	0.7922 (Large)

The Wilcoxon signed-rank test revealed statistically significant differences between token counts before and after optimization for all three strategies ($p < 0.001$). Therefore, the null hypothesis of no difference was rejected in every case. The confidence intervals for the pseudo-medians did not include zero, providing additional evidence that the optimization strategies consistently reduced token usage.

Among the evaluated methods, Code Formatting exhibited the largest median reduction, with an estimated pseudo-median of approximately 34 tokens. Verbose Removal achieved a pseudo-median reduction of 26 tokens, while Structured Concise produced a pseudo-median reduction of approximately 20.5 tokens. Although the magnitude of reduction varied across strategies, all approaches demonstrated substantial improvements in token efficiency.

The calculated effect sizes further indicate that the practical impact of the optimization strategies was considerable. According to commonly accepted interpretation guidelines, effect size values greater than 0.50 represent large effects. All three strategies exceeded this threshold, with effect sizes ranging from 0.7922 to 0.8684. Verbose removal and Code formatting produced the largest practical effects ($r = 0.8684$), while Structured Concise also demonstrated a strong effect ($r = 0.7922$).

Overall, the statistical analysis confirms that each prompt optimization strategy produced significant and practically meaningful reductions in token usage. The combination of highly significant Wilcoxon test results and large effect sizes provides strong evidence that the proposed optimization methods are effective for improving prompt efficiency across the evaluated dataset.

4.5 Quality Assessment Results

The table below summarizes the rating distribution across all three optimization strategies, each evaluated against 50 prompt pairs drawn from the 300-prompt quality assessment sample.

Results indicate that all three optimization strategies preserved or improved response quality, with no downgraded outputs observed across the 150 evaluated prompt pairs. Verbose Removal achieved the highest number of quality improvements, with 15 upgraded responses (30%), followed by Structured Concise with 10 upgrades (20%). Code Formatting maintained response quality in all cases, producing 50 satisfactory outcomes without any upgrades or downgrades. Overall, 25 responses (16.7%) were upgraded, while 125 responses (83.3%) remained satisfac-

Table 9: Quality Assessment Rating Distribution by Strategy

Optimization Strategy	Upgraded	Satisfactory	Downgraded	Total
Verbose Removal	15	35	0	50
Structured Concise	10	40	0	50
Code Formatting	0	50	0	50
Total	25	125	0	150

tory, demonstrating that the optimization strategies consistently maintained output quality while occasionally enhancing it.

Semantic equivalence was further verified using BERTScore F1 computed via roberta-large across all 150 paired prompt comparisons. Results are summarized in Table 10. All 150 pairs exceeded the equivalence threshold of F1 $\geq$ 0.85 (100% compliance). The overall mean F1 was 0.9625 (SD = 0.0385), with mean precision of 0.9736 and mean recall of 0.9521, confirming that optimized prompts preserved the semantic content of their originals with high fidelity. The Upgraded group (n = 25) achieved a mean F1 of 0.9399 (SD = 0.0422), while the Satisfactory group (n = 125) achieved a mean F1 of 0.9670 (SD = 0.0363). The slightly lower mean in the Upgraded group is consistent with those prompts having undergone more substantive structural restructuring, yet all remained well above the equivalence threshold. Taken together, the output-quality ratings and BERTScore results confirm that the optimization strategies were both semantically conservative and quality-preserving.

Table 10: Semantic Equivalence Evaluation — BERTScore F1 (roberta-large)

Group	Count	Mean F1	Std Dev	Min F1	Max F1	% $\geq$ 0.85	Equivalence
Upgraded	25	0.9399	0.0422	0.8887	0.9886	100.0%	Full
Satisfactory	125	0.9670	0.0363	0.8833	0.9968	100.0%	Full
Overall	150	0.9625	0.0385	0.8833	0.9968	100.0%	Full

5 Discussion

5.1 Strategy Effectiveness Analysis

5.1.1 Verbose Removal

The Verbose Removal strategy achieved a mean token reduction of 51.98% across all 160 tested prompts and was identified as the most reliable optimization approach overall, achieving a 100% success rate in producing token reduction across all tested cases. The results indicate that systematic removal of redundant and polite linguistic structures can significantly improve prompt efficiency while preserving semantic intent. The most common verbose patterns identified, along with their replacement rules, frequency of occurrence, and average token savings, are summarized in Table 11. These patterns were primarily associated with redundant politeness markers and extended request formulations. Transformations such as converting "due to the fact that" to "since" and "could you please examine" to "examine" demonstrated the highest efficiency gains, while still maintaining clarity of meaning. However, one case ("I need help writing $\rightarrow$ write") resulted in a negative saving, highlighting that excessive compression may occasionally introduce inefficiency due to loss of contextual clarity or downstream rephrasing.

Overall, these findings demonstrate that verbose pattern elimination aligns strongly with

Table 11: Most Common Verbose Patterns and Token Savings

Pattern	Replacement	Frequency	Average Token Saving
Please review this	Review	51	23.8
Due to the fact that	since	34	28.4
Kindly use	Use	17	25.4
Please include	Include	17	21.7
Could you please generate	Generate	17	26.1
Please explain	Explain	17	27.4
I would like you to create	Create	17	20.6
Kindly include	Include	17	20.6
Could you please examine	Examine	17	29.2
I need help writing	Write	17	-9.4
I'm debugging	Debug	17	13.8
Can you explain	Explain	17	25.4

concise communication principles, where redundancy is reduced without compromising clarity. The results further suggest that rule-based transformations, particularly those implemented via regular expression matching, are highly effective for scalable prompt optimization systems.

5.1.2 Structured Concise Format

The Structured Concise strategy demonstrated a substantial but more variable impact on token efficiency, achieving an overall average reduction of 33.36% across 160 prompts. Compared to the Verbose Removal strategy, which produced higher and more consistent reductions, Structured Concise optimization exhibited significantly greater variability, as reflected in a standard deviation of 20.71. This indicates that the effectiveness of the strategy is strongly dependent on the structure and complexity of the initial prompt.

The strategy works by transforming verbose narrative prompts into structured imperative formats, typically replacing descriptive or explanatory prose with concise bullet-style instructions. This restructuring removes transitional language, redundant framing, and unnecessary elaboration while preserving all essential task components. As a result, the method is particularly effective for multi-part and procedural prompts that contain layered instructions or extended narrative context.

However, the observed variability suggests that performance is not uniform across all prompt types. Prompts that are already moderately structured tend to show limited improvement, while highly narrative or unstructured inputs yield significantly larger gains. This sensitivity to baseline structure explains the wide dispersion in results, despite strong average performance. Overall, Structured Concise optimization is most suitable for complex, multi-step prompts rather than uniformly applied across all input types.

5.1.3 Code Formatting Optimization

Code Formatting optimization showed moderate but inconsistent improvements, achieving an overall average reduction of 29.71% across 160 prompts. However, performance at the individual test level was highly uneven, with many cases showing little to no effective reduction in real-world API execution conditions. This outcome suggests that most code-related prompts are already relatively concise and structurally optimized at the baseline, leaving limited scope for additional compression. In many cases, prompts consist primarily of direct code instructions or technical specifications, which do not contain the types of redundant linguistic structures targeted by this optimization strategy.

The strategy was most effective in specific scenarios where prompts included repetitive introductions to multiple code blocks, unstandardized instructional phrasing, or verbose educational explanations accompanying code. In such cases, removing redundant framing and standardizing instruction formats produced measurable token savings. However, across general-purpose coding prompts, the improvements were minimal. Overall, Code Formatting optimization should be considered a domain-selective strategy, best suited for structured code documentation, educational programming content, and automated code review systems, rather than a universal optimization method. This domain-selectivity is a finding about prompt-type applicability, not a reflection of lower experimental rigour; the strategy was tested on the same number of prompts (160) as the other two strategies and its performance is comparably well-characterized within its target domain.

Table 12: Strategy Effectiveness Comparison

Strategy	Avg Reduction (%)	Best Single Result(%)	Model	Reliability
Verbose Removal	51.98	65.6%	gemma-3-27b-it	100% positive
Structured Concise	33.36	59.7%	gemma-3-27b-it	Variable
Code Formatting	29.71	60.9%	gemma-3-27b-it	Domain-specific

5.2 Qualitative Findings

Beyond quantitative token reduction, several qualitative observations emerged from the testing process. First, all strategies maintained prompt semantic content; the AI-judge quality assessment indicated that none of the 150 evaluated prompt pairs produced a downgraded output, and BERTScore semantic equivalence evaluation confirmed that all 150 pairs exceeded the F1 ≥ 0.85 threshold (mean F1 = 0.9625), providing convergent evidence that both output quality and prompt-level communicative intent were preserved across strategies. Second, the Structured Concise format frequently improved prompt clarity as a byproduct of optimization, reducing ambiguity and in several cases improving response quality. Third, while not directly measured in this study, optimized prompts are expected to process faster due to reduced context requirements, an effect documented in prior work by Han et al. [11]. Fourth, strategy interactions were observed: Verbose Removal and Structured Concise complement one another.

These qualitative observations are further corroborated by the structured quality assessment conducted on a sample of 300 prompts drawn from the full 960-prompt dataset. Across all 150 paired comparisons, no optimization strategy produced any degraded output. Verbose Removal yielded the highest upgrade rate at 30%, consistent with its superior quantitative performance. Structured Concise achieved a 20% upgrade rate while Code Formatting produced a neutral effect with all 50 pairs rated Satisfactory. These findings confirm that the optimization strategies are non-destructive and, in the case of Verbose Removal and Structured Concise, demonstrably beneficial to output quality.

5.3 Practical Implementation Guide

5.3.1 Verbose Removal Implementation

The Verbose Removal strategy is the strongest candidate as a first-pass optimization for natural-language prompts, owing to its consistency across prompt types, achieving a 51.98% average reduction with a standard deviation of 7.01. For code-heavy prompts, Code Formatting should be considered first; for multi-part narrative prompts, Structured Concise is the more appropriate

primary strategy. Implementation proceeds in three steps. In Step 1, verbose patterns are identified in existing prompts using regex or natural language processing, targeting politeness markers ("please," "kindly," "appreciate"), hedging constructions ("would you like," "I was wondering"), wordy prepositional phrases ("in order to" replaced by "to"; "due to the fact that" replaced by "because"), and temporal padding ("at this point in time" replaced by "now"). In Step 2, automated pattern-based replacement is applied using regular expressions. In Step 3, semantic validation is performed to verify that optimized prompts maintain meaning, through manual review for critical prompts, automated similarity scoring (cosine similarity, BLEU), or A/B testing for production applications.

5.3.2 Structured Concise Implementation

Structured concise formatting is effective for verbose prose prompts but exhibits substantial variability in performance, achieving an average token reduction of 33.36% with a standard deviation of 20.71. It is best applied selectively to multi-part requests with narrative descriptions, prompts containing embedded questions or requirements, documentation or specification requests, and analysis tasks involving multiple sub-questions. The implementation approach involves extracting instructions from narrative text, converting them into imperative bullet points, removing transitional phrases, consolidating redundant instructions, and preserving code blocks and data unchanged. This strategy should be avoided for already concise prompts, creative writing tasks, and conversational prompts where formal structure may be inappropriate.

5.3.3 Integration with LLM Applications

Organizations can integrate token optimization into LLM workflows through a pre-processing pipeline of the form: User Prompt, then Strategy Selection (based on prompt type: Verbose Removal for natural-language prompts, Code Formatting for code-heavy prompts, Structured Concise for multi-part narrative prompts), then Semantic Validation, then Optimized Prompt, then LLM API call. The Verbose Removal stage is shown first in this example as it applies most broadly across the dataset tested here, but it is not the only or universally preferred stage. For real-time optimization, the strategy should be applied before each API call, with caching of optimized prompts for repeated use, logging of optimization metrics for monitoring, and fallback to the original prompt if optimization introduces errors. For batch processing, prompt libraries and templates can be optimized offline, with A/B testing used to validate optimized variants against originals and optimization guidelines established for prompt engineers.

5.4 Cost-Benefit Analysis

The economic implications of token optimization are substantial at scale. At a 51.98% average reduction from Verbose Removal, an organization spending $10,000 per month on LLM APIs could realize savings of approximately $5,198 per month, or $62,376 annually. Implementation costs are estimated at 40-80 development hours for optimization pipeline integration, 4-8 hours per month for maintenance and monitoring, and 8-16 hours per month for quality assurance and validation. Break-even is typically reached within one to three months depending on deployment scale, with savings compounding as LLM usage grows. The consistent tokenization behavior observed across the 11 Gemini and 2 Gemma 4 model variants (13 in total, excluding Gemma 3) under a shared fixed prompt set suggests that a single optimization pipeline may be applicable across the Gemini model family without per-model tuning, though this reflects within-ecosystem tokenizer consistency rather than independent cross-model validation.

5.5 Industry Applications

Different industries can apply token optimization strategies in ways tailored to their primary use cases. In software development, code review prompts benefit from Verbose Removal (51.98% reduction) and documentation generation benefits from Structured Concise (33.36% reduction). In data science, statistical analysis requests and visualization recommendations yield 51.98% and 33.36% reductions respectively through Verbose Removal and Structured Concise. In content creation, technical writing is well-suited to Verbose Removal (51.98% reduction), while creative writing prompts are less amenable to optimization and may require manual judgment. In customer support, FAQ generation and knowledge base creation yield 51.98% reductions through Verbose Removal. In research and academia, literature review assistance and methodology explanation tasks are particularly responsive to Verbose Removal and Structured Concise, with reductions in the 51.98% and 33.36% range.

6 Conclusion

This research established a systematic framework for LLM token optimization through comprehensive testing with random prompts using actual Gemini API token counting. By testing 480 diverse prompts across three optimization strategies and measuring real token consumption via the Gemini countTokens endpoint across 16 different models, this study validated that simple prompt engineering techniques can significantly reduce token costs within the Gemini ecosystem, without requiring learned or search-based compression models, and while maintaining response quality within the tested prompt set. Whether these gains extend to other LLM families or compare favorably against established compression baselines remains a direction for future work.

6.1 Key Research Findings

Six principal findings emerged from this study. First, comprehensive API-based evaluation across 480 prompt executions demonstrated substantial reductions in token usage, yielding an overall mean reduction of approximately 37.15% across Gemini-family models (excluding Gemma 3, which achieved 43.55%), computed as the average token reduction under a shared fixed prompt set across the Gemini 2.0, 2.5, 3 preview, latest, and Gemma 4 model groups (Table 4). Among the individual methods, Verbose Removal achieved the highest mean reduction of 51.98% based on 160 prompts, followed by Structured Concise at 33.36% based on 160 prompts and Code Formatting at 29.71% based on 160 prompts, with a maximum single-instance reduction of 65.6% observed in highly verbose prompts. Second, cross-model evaluation across 16 large language model variants spanning the Gemini 2.0, 2.5, 3 preview, latest Gemini series, and Gemma 3 to Gemma 4 families indicated that prompt optimization effects were broadly consistent across model architectures. Most Gemini-family models exhibited an identical mean reduction of 37.15%, as expected given the shared fixed prompt set used across all models, while Gemma 3 models achieved a higher mean reduction of 43.55%, suggesting moderate sensitivity to tokenizer or model-family characteristics rather than complete invariance. Third, Verbose Removal emerged as the most effective and stable optimization strategy, combining the highest mean reduction with the lowest variability, with a standard deviation of 7.01, and exhibiting no observed degradation cases across evaluations. Fourth, inferential statistical analysis using the Wilcoxon signed-rank test confirmed that all observed reductions were highly statistically significant with p values less than 2.2×10^{-16}, and effect sizes ranging from 0.7922 to 0.8684, thereby providing strong within-sample evidence against the null hypothesis of no difference between original and optimized prompts within the tested Gemini ecosystem; these tests do not extend to cross-strategy comparisons or to other LLM families, and cross-ecosystem generalizability remains to be established. Fifth, qualitative assessment of output preservation across

150 evaluated prompt pairs indicated that optimization did not compromise response quality, with 25 cases rated as improved, 125 cases rated as satisfactory, and none rated as degraded. Finally, all measurements were derived using production-grade Gemini countTokens API endpoints for each model, ensuring that token counts reflect real-world deployment conditions and maintain high experimental reproducibility and external validity.

6.2 Practical Implications

For individual practitioners working with LLMs, this research yields several actionable recommendations. Verbose Removal is the strongest first-pass strategy for natural-language prompts, given its 51.98% average reduction across 160 API-validated tests, its low variability (standard deviation 7.01), and its 100% positive-outcome rate within the tested prompt set. Its consistent performance across the 11 Gemini and 2 Gemma 4 model variants tested (13 in total, excluding Gemma 3)—a result of shared tokenization behaviour within the Gemini ecosystem rather than independent cross-model replication—suggests no per-model tuning is required within this model family. Structured Concise formatting should be applied selectively to multi-part narrative prompts, where it achieved a 33.36% mean reduction; however, its high variability (standard deviation 20.71) means results differ substantially across prompt types, and it should be avoided for already concise or conversational prompts. Code Formatting optimization is best applied to domain-specific scenarios—structured code documentation, educational content, and automated code review—where it can deliver meaningful savings; it is not recommended as a general-purpose pipeline component given the high proportion of code prompts that are already compact and yield little additional reduction.

For organizations scaling LLM deployments, the findings support a clear economic case for optimization. The overall mean token reduction of 37.15% across Gemini-family models (Table 4), and 51.98% with Verbose Removal specifically, translate directly to proportional API cost savings. (Note: the figure of 28.3 reported in Table 5 represents the average absolute token saving per prompt per model, not a percentage reduction, and should not be compared directly with the strategy-level percentages.) The identical summary statistics observed across the 11 Gemini and 2 Gemma 4 model variants (13 in total, excluding Gemma 3) reflect consistent tokenization behaviour within the Gemini family under a shared prompt set; this supports the practical expectation that a single optimization pipeline will not require per-model tuning within this ecosystem, though it does not constitute independent cross-model replication. Secondary benefits include improved processing latency from reduced context size, reduced energy consumption and associated environmental benefits as documented by Luccioni et al. [14], and increased accessibility of LLM capabilities for resource-constrained environments.

6.3 Future Research Directions

Ten avenues for future research emerge from this work. First, automated optimization tools that integrate directly with LLM APIs could provide real-time optimization suggestions or automatic prompt refinement, potentially implemented as browser extensions or IDE plugins. Second, systematic study of how optimization affects response quality across different models is needed; while token reduction is valuable, controlled A/B testing comparing optimized and original prompts should establish quality preservation thresholds. Third, domain-specific optimization patterns warrant investigation for specialized fields including legal analysis, medical documentation, and creative writing, each of which may resist or reward optimization differently. Fourth, dynamic optimization strategies that adapt in real time to remaining token budgets and conversation context could address multi-turn scenarios not examined in this study. Fifth, cross-model validation with OpenAI GPT, Anthropic Claude, Meta Llama, and other architectures would test whether the model-agnostic finding generalizes beyond the Gemini ecosystem. Sixth, the interaction effects of applying multiple strategies sequentially, for example Verbose Removal

followed by Structured Concise, should be systematically studied. Seventh, extension of optimization strategies to languages beyond English is important given that different languages have distinct verbose patterns, politeness conventions, and grammatical structures. Eighth, development of metrics to predict optimization effectiveness from prompt characteristics such as length, syntactic complexity, and semantic density would enable automated routing of prompts to the most appropriate strategy. Ninth, user interface research should examine how optimization suggestions are best presented to users and whether automatic application or interactive suggestions better serve different user populations. Tenth, longitudinal studies could investigate whether users who receive optimization feedback learn to compose more concise prompts naturally over time.

6.4 Research Limitations

The study has several limitations worth acknowledging. Testing was confined to three optimization strategies without exploring potential combinations or hybrid approaches, and the randomly generated prompts may not fully represent specialized domains like legal, medical, or creative writing contexts where optimization dynamics could differ significantly. Additionally, the quality assessment covered only 150 of the 480 unique prompts (31.3%) and relied on a single automated judge (DeepSeek V4 Flash, open source, accessed 26 April 2026) without inter-rater reliability validation. The assessment also included a BERTScore semantic equivalence evaluation (Section 3.6.5), confirming that all 150 paired prompts exceeded the F1 $\geq$ 0.85 equivalence threshold (overall mean F1 = 0.9625), addressing the concern that phrasing changes might affect LLM comprehension in ways not captured by output-quality ratings alone. The analysis was also limited to static, single-turn prompts, leaving multi-turn conversational dynamics unexplored.

Validation was conducted exclusively within the Gemini API ecosystem, which, while offering confirmation across 16 model variants, limits the generalizability of findings to other LLM families such as GPT, Claude, or Llama. A further practical constraint was the Gemini Free API's token limits, which restricted the volume and length of prompts that could be tested, preventing large-scale or long-form prompt evaluation. Cross-ecosystem testing and expanded prompt datasets in future research would substantially strengthen these findings. Additionally, the multi-model evaluation used the same fixed set of 30 prompts for every model rather than independently sampled subsets per model. This design choice explains the identical summary statistics (mean, standard deviation, minimum, and maximum) observed across the 11 Gemini and 2 Gemma 4 models (13 in total; Gemma 3 is excluded, as it shows a distinct, higher-reduction pattern) in Table 6 and confirms tokenizer consistency rather than constituting independent cross-model replication. Future work should evaluate independently sampled prompt sets per model to more rigorously test cross-model robustness. Finally, no direct comparison against established prompt-compression baselines or human-engineered prompts was included. The study does not benchmark its three rule-based strategies against published compression methods (e.g., LLMLingua, Selective Context, or AutoCompressor) or against manually optimized prompt sets. This limits the ability to situate the magnitude of the reported gains relative to the broader literature. Including such comparisons is identified as a priority direction for future work.

References

1 T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, and D. Amodei, "Language models are few-shot learners," arXiv preprint arXiv:2005.14165, May 2020.

2 T. Kudo, "Subword regularization: Improving neural network translation models with multiple subword candidates," in Proc. 56th Annu. Meeting Assoc. Comput. Linguistics, Melbourne, Australia, 2018, pp. 66–75.

3 J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling laws for neural language models," arXiv preprint arXiv:2001.08361, Jan. 2020.

4 J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, and W. Fedus, "Chain-of-thought prompting elicits reasoning in large language models," Adv. Neural Inf. Process. Syst., vol. 35, pp. 24824–24837, 2022.

5 P. Liang, R. Bommasani, T. Lee, D. Tsipras, D. Soylu, M. Yasunaga, Y. Zhang, D. Narayanan, Y. Wu, A. Kumar, B. Newman, L. Yuan, B. Yanma, C. Zhang, C. Cosgrove, C. D. Manning, and C. Re, "Holistic evaluation of language models," Trans. Assoc. Comput. Linguistics, vol. 10, pp. 100–123, 2022.

6 H. Puerto, M. Tutek, S. Aditya, X. Zhu, and I. Gurevych, "Code Prompting Elicits Conditional Reasoning Abilities in Text+Code LLMs," arXiv preprint arXiv:2401.10065, Jan. 2024.

7 S. Min, X. Lyu, A. Holtzman, M. Artetxe, M. Lewis, H. Hajishirzi, and L. Zettlemoyer, "Rethinking the role of demonstrations: What makes in-context learning work?" arXiv preprint arXiv:2202.12837, Feb. 2022.

8 R. M. G. Alarcia, "Optimizing Token Usage on Large Language Model Conversations Using the Design Structure Matrix," arXiv preprint arXiv:2410.00749, Oct. 2024.

9 Google, "Prompt design strategies: Gemini API," Google AI for Developers, 2024. [Online]. Available: google.dev

10 M. Aubakirova, A. Atallah, C. Clark, J. Summerville, and A. Midha, "State of AI: An empirical 100 trillion token study with OpenRouter," arXiv preprint arXiv:2601.10088, Jan. 2026.

11 T. Han, Z. Wang, C. Fang, S. Zhao, S. Ma, and Z. Chen, "Token-Budget-Aware LLM Reasoning," arXiv preprint arXiv:2412.18547, 2025. Available: arxiv.org

12 T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," Adv. Neural Inf. Process. Syst., vol. 35, pp. 22199–22213, 2022.

13 J. Hu, W. Zheng, Y. Liu, and Y. Liu, "Optimizing Token Consumption in LLMs: A Nano Surge Approach for Code Reasoning Efficiency," arXiv preprint arXiv:2504.15989, 2025. Available: arxiv.org

14 A. S. Luccioni, S. Viguier, and A.-L. Ligozat, "Estimating the carbon footprint of BLOOM, a 176B parameter language model," J. Mach. Learn. Res., vol. 24, no. 253, pp. 1–15, 2023.

Declarations

Funding Statement

No funding received.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Ethical Considerations:

This study is a comparative theoretical analysis using previously published results. It does not involve human participants, animals, or personal data; therefore, institutional ethical approval was not required. Figure provenance and permissions: Figure 1 is an original schematic illustration created by the authors specifically for this manuscript. It is not reproduced, adapted, or modified from any copyrighted source.

AI Usage Statement

AI tools were used for code generation, prompt evaluation, language refinement, and draft structuring support. All scientific claims, benchmark values, interpretations, and reference mapping were verified and curated by the author.