ADAPT: A Position Paper on Adaptive Token Allocation in AI Prompting Based on Input Complexity

Adithya Srivatsa
Independent Researcher
archlinuxadithya@gmail.com

July 19, 2025

Abstract

Large Language Models like GPT-4 have token limits that require careful consideration of prompt structure. Current prompt engineering approaches typically use fixed token allocation strategies across all inputs within a task - for example, always dedicating 40% of tokens to instructions, 40% to examples, and 20% to context, regardless of input complexity.

I argue that this one-size-fits-all approach is fundamentally suboptimal. A simple math problem like "What is 2+2?" likely requires a different prompt structure than a complex calculus word problem, even though both are "math tasks." Simple inputs may need minimal instructions but clear formatting guidance, while complex inputs might benefit from detailed examples and reasoning chains.

This position paper identifies a significant gap in current prompt engineering research and proposes ADAPT, a framework for dynamically allocating prompt tokens based on individual input characteristics. I outline the key technical challenges, propose potential solutions, and present a research agenda for validating this approach. Rather than presenting experimental results, I focus on making a compelling case that adaptive token allocation represents an important and under-explored direction for improving AI system efficiency.

I believe this research direction could lead to substantial improvements in language model performance across diverse tasks, and I call for the research community to investigate these ideas with the computational resources necessary for proper validation.

Keywords: Large Language Models, Prompt Engineering, Token Optimization, Adaptive Systems, Position Paper

1 Introduction

1.1 The Overlooked Problem in Prompt Engineering

As a high school student experimenting with language models, I've observed a fundamental inefficiency in how current prompt engineering approaches work. Despite rapid advances in techniques like chain-of-thought prompting [Wei et al., 2022] and automated prompt optimization [Shin et al., 2020], virtually all existing methods treat prompt structure as a task-level decision rather than an input-level one.

Consider these two math problems:

- **Simple**: "What is 15 + 27?"
- **Complex**: "A garden has length 3 meters more than twice its width. If the perimeter is 36 meters, what is the area?"

[0]This position paper is written by an independent high school student exploring machine learning and prompt engineering. The work presents a theoretical framework and research agenda that would require substantial future validation by researchers with greater computational resources.

Current systems would use identical prompt structures for both, perhaps allocating 40% of tokens to instructions, 40% to examples, and 20% to context. But this seems wasteful - the simple problem needs clear format specification but minimal examples, while the complex problem would benefit from detailed step-by-step examples but could use shorter instructions.

1.2 Why This Matters Now: Economic and Performance Implications

This problem is becoming increasingly important for several reasons supported by current market data:

Token Economics: The financial impact of inefficient prompting is substantial. OpenAI's current pricing structure charges $0.06 per 1K prompt tokens for GPT-4, while GPT-4o mini costs 15 cents per million input tokens. For systems processing millions of queries daily, unnecessary tokens in prompts represent significant cost overhead. A 20% reduction in average prompt length through adaptive allocation could translate to thousands of dollars in monthly savings for enterprise applications.

Context Window Constraints: Even with expanding context windows, optimal token allocation remains crucial. More efficient prompting means more room for actual task content.

Performance Gaps: Recent research indicates that prompt structure significantly affects model performance. Chain-of-thought prompting can produce striking empirical gains on arithmetic and reasoning tasks, but these gains are likely dependent on input complexity. Simple inputs may actually suffer from over-prompting, while complex inputs benefit from structured reasoning guidance.

Real-world Implementation Challenges: Companies deploying LLM-based systems face the challenge of balancing cost, performance, and latency. Current approaches require either expensive over-prompting of all inputs or accepting performance degradation on complex cases.

1.3 Evidence from Industry Practice

Several observations from current industry practice support the need for adaptive allocation:

Manual Prompt Tuning: Many practitioners manually create different prompt templates for different input types within the same task category. This suggests an intuitive understanding that optimal prompting varies by input characteristics, but current frameworks don't systematically address this need.

Cost-Performance Trade-offs: Organizations often use simpler, shorter prompts to reduce costs, potentially sacrificing performance on complex inputs. Adaptive allocation could optimize this trade-off automatically.

Prompt Engineering Complexity: The time and expertise required for manual prompt optimization scales poorly. Automated adaptive systems could democratize effective prompting.

1.4 My Contribution

This position paper makes several contributions to prompt engineering research:

1. **Problem Identification**: I formally identify the input-agnostic allocation problem in current prompt engineering

2. **Framework Proposal**: I propose ADAPT, a systematic approach to adaptive token allocation

3. **Technical Roadmap**: I outline specific algorithms and complexity measures that could enable this approach

4. **Research Agenda**: I present concrete research questions and experimental protocols for validating these ideas
5. **Gap Analysis**: I show why existing prompt optimization methods don't address this problem
6. **Economic Analysis**: I quantify the potential cost savings and performance improvements from adaptive allocation

Rather than claiming to have solved this problem, I aim to convince the research community that it's worth solving.

1.5 Development Timeline and AI Collaboration

This position paper was developed over a single day as a rapid exploration of an observed gap in prompt engineering research. The use of AI tools (ChatGPT, Claude AI, Qwen, DeepSeek) was instrumental in enabling this compressed timeline - what might traditionally require weeks of literature review, technical writing, and formatting was accomplished in 24 hours through AI-assisted research and articulation. While the time constraint necessarily limited the depth of technical development, I believe the core insight merits investigation by researchers with greater resources and time. This demonstrates both the potential of AI-augmented research for rapid ideation and the continued importance of human insight in identifying fundamental problems that automated systems might overlook.

2 Related Work and the Gap I Identify

2.1 Current Prompt Engineering Approaches

Modern prompt engineering has produced several powerful techniques:

Chain-of-Thought Prompting [Wei et al., 2022] improves reasoning by asking models to show intermediate steps. Experiments on three large language models show that chain of thought prompting improves performance on a range of arithmetic, commonsense, and symbolic reasoning tasks. However, all inputs in a task receive the same "show your work" instruction, regardless of whether they need it.

Few-Shot Learning provides examples to help models understand tasks. But the same examples are shown for every input, even when some inputs are much simpler than the examples.

Instruction Following trains models to follow detailed task descriptions. Yet every input gets the same level of instruction detail.

2.2 Automated Prompt Optimization

Recent work has begun automating prompt design:

DSPy [Khattab et al., 2023] provides frameworks for prompt programming but still operates at the task level - it finds one good prompt structure per task.

AutoPrompt [Shin et al., 2020] automatically searches for effective prompt templates using gradient-based methods. AutoPrompt develops an automated method to create prompts for a diverse set of tasks, based on a gradient-guided search. However, it seeks a single template per task rather than adapting to individual inputs.

Grid Search Approaches try different combinations of instructions, examples, and context allocation, but select one combination to use universally.

2.3 Emerging Research on Input-Specific Adaptation

While most work focuses on task-level optimization, some recent research hints at the importance of input-level adaptation:

Active Prompting: Recent work explores selecting better examples for chain-of-thought prompting based on uncertainty estimation, suggesting that different inputs may benefit from different supporting materials.

Dynamic Few-Shot Selection: Some systems now select different examples for different test inputs, indicating recognition that one-size-fits-all example selection is suboptimal.

However, these approaches focus on example selection rather than comprehensive token allocation across all prompt components.

2.4 The Fundamental Gap

All existing approaches share a common assumption: **that the optimal prompt structure is a property of the task, not the individual input**.

This assumption leads to several problems:

Over-prompting Simple Inputs: Easy inputs get overwhelmed with unnecessary examples and verbose instructions, potentially confusing the model and wasting tokens.

Under-prompting Complex Inputs: Difficult inputs don't receive sufficient guidance, examples, or reasoning chains to perform well.

Inefficient Token Usage: Tokens are wasted on guidance that isn't needed for each specific input.

Missed Performance Opportunities: Models could perform better if prompts were tailored to input difficulty.

To my knowledge, no existing work has systematically addressed comprehensive input-level prompt adaptation across all prompt components, making this an important gap in the literature.

3 The ADAPT Framework: A Research Proposal

3.1 Core Insight and Approach

I propose that optimal prompting should be a function of both task type AND individual input characteristics. The ADAPT (Adaptive Dynamic Allocation for Prompt Tokens) framework would analyze each input and dynamically allocate tokens based on predicted complexity and requirements.

3.2 Technical Components

3.2.1 Input Complexity Assessment

The first challenge is measuring how "complex" an input is relative to its task. I propose a multi-dimensional approach:

Semantic Complexity: Using sentence embeddings to measure how different an input is from typical task examples. Inputs that are outliers in the semantic space likely require more guidance.

Surface-Level Features: Analyzing measurable characteristics:

- Text length and sentence count
- Reading level (Flesch-Kincaid, etc.)
- Named entity density

- Syntactic complexity measures

Task-Specific Indicators: Different domains would need specialized complexity measures:

- Math: Number of operations, equation types, word problem vs. symbolic
- Reading: Passage length, question type, inference requirements
- Code: Function count, algorithmic complexity, language features

These would combine into a complexity score:

$$\text{complexity}(x) = \alpha \cdot \text{semantic}(x) + \beta \cdot \text{surface}(x) + \gamma \cdot \text{task\_specific}(x) \tag{1}$$

where the weights α, β, γ would be learned from data.

3.2.2 Hierarchical Token Allocation

Rather than simple three-way splits, I propose hierarchical allocation:

Level 1 - Component Allocation: Decide tokens for Instructions (I), Examples (E), and Context (C)

$$\text{total\_tokens} = \text{tokens}_I + \text{tokens}_E + \text{tokens}_C + \text{tokens}_{\text{input}} \tag{2}$$

Level 2 - Sub-component Allocation: Within each component:

- Instructions: Task description, output format, constraints, reasoning guidance
- Examples: Positive examples, negative examples, edge cases, step-by-step solutions
- Context: Background knowledge, relevant facts, domain-specific information

This gives fine-grained control with 9+ allocation parameters.

3.2.3 Learning Algorithm Framework

I propose treating allocation as a contextual bandit problem with economic constraints. The system would optimize for both performance and cost efficiency:

Algorithm 1 ADAPT Learning Framework (Proposed)

```
1: Initialize allocation strategies $S = \{s_1, s_2, ..., s_k\}$
2: Initialize performance and cost models
3: for each new input $x$ do
4:    Compute complexity features $f(x)$
5:    Compute cost constraint $c_{max}$ (budget-dependent)
6:    For each strategy $s_i$, compute:
7:       $\text{performance\_score}(s_i) = \mathbb{E}[\text{accuracy}|s_i, f(x)]$
8:       $\text{cost\_score}(s_i) = \text{token\_cost}(s_i)$
9:       $\text{utility}(s_i) = \text{performance\_score}(s_i) - \lambda \cdot \text{cost\_score}(s_i)$
10:   Select strategy $s^* = \arg\max_i \text{utility}(s_i)$ subject to cost constraint
11:   Apply allocation $s^*$, observe performance $r$ and actual cost $c$
12:   Update models using $(f(x), s^*, r, c)$
13: end for
```

The parameter λ allows tuning the performance-cost trade-off based on application requirements and budget constraints.

4 Research Agenda: Key Questions and Experimental Protocols

Rather than presenting speculative results, I outline the key research questions that need investigation and propose experimental protocols for addressing them.

4.1 Fundamental Research Questions

RQ1: Complexity Measurement How can input complexity be effectively measured across different task types? Which combination of semantic, surface-level, and task-specific features best predicts optimal allocation?

RQ2: Allocation Granularity What level of allocation granularity provides the best performance-complexity tradeoff? Is hierarchical allocation worth the additional complexity?

RQ3: Learning Efficiency How quickly can the system learn effective allocation strategies? How much training data is needed to surpass fixed allocation baselines?

RQ4: Economic Optimization Can adaptive allocation achieve better performance-cost Pareto frontiers than fixed strategies? What are the trade-off curves?

RQ5: Generalization Do learned allocation strategies generalize across different models, task variations, and domains?

RQ6: Performance Bounds What are the theoretical and practical limits of performance improvement from adaptive allocation?

4.2 Proposed Experimental Protocol

Phase 1: Complexity Measure Validation

- Collect human judgments of input complexity across multiple task types
- Develop and validate automated complexity measures
- Establish that complexity correlates with optimal allocation patterns
- Create baseline cost-performance profiles for different allocation strategies

Phase 2: Small-Scale Allocation Studies

- Manually design allocation strategies for inputs of different complexity
- Test performance across complexity levels with controlled experiments
- Establish upper bounds for potential improvement
- Quantify cost savings from avoiding over-prompting simple inputs

Phase 3: Learning Algorithm Development

- Implement and test different learning approaches (bandits, reinforcement learning, etc.)
- Compare learning efficiency and final performance
- Study generalization across tasks and models
- Evaluate economic optimization capabilities

Phase 4: Large-Scale Validation

- Test across diverse NLP benchmarks
- Compare against state-of-the-art prompt optimization methods
- Analyze computational overhead and practical deployment considerations
- Conduct cost-benefit analysis for enterprise deployment scenarios

4.3 Success Criteria

I propose that this research direction would be considered successful if:

- Adaptive allocation consistently outperforms fixed allocation across multiple tasks
- Improvements are statistically significant and practically meaningful (¿5% on average)
- Cost efficiency improvements are demonstrated (¿15% token reduction with maintained performance)
- The approach generalizes across different language models
- Computational overhead is acceptable for practical deployment (¡10% latency increase)
- The system learns effective strategies with reasonable amounts of training data (¡1000 examples per task)

5 Why This Research Direction Matters

5.1 Theoretical Significance

This work would advance understanding of several important questions in AI:

Optimal Prompting Theory: What makes a prompt effective, and how does this depend on input characteristics?

Adaptive AI Systems: How can AI systems automatically adjust their behavior based on input analysis?

Resource Optimization: How can AI systems be made more efficient without sacrificing performance?

Input-Output Complexity Relationships: How does the complexity of desired outputs relate to optimal input structuring?

5.2 Practical Impact

If successful, this research could have significant practical benefits:

Cost Reduction: With GPT-4 costing $0.06 per 1K prompt tokens and millions of daily queries in enterprise applications, a 20% reduction in average prompt length could save thousands of dollars monthly per application.

Performance Improvement: Better prompt matching could improve accuracy across diverse applications, particularly benefiting complex reasoning tasks that currently suffer from under-prompting.

Accessibility: More efficient prompting could make advanced AI capabilities accessible with smaller models or budgets, democratizing AI access.

Automation: Reducing the need for manual prompt engineering could accelerate AI adoption across industries that lack specialized expertise.

Scalability: Automated optimization would enable personalized AI applications at scale, where manual optimization is impractical.

5.3 Broader Implications

This work connects to several broader trends in AI research:

Personalization: Moving from one-size-fits-all to individualized approaches.

Efficiency: Making AI systems more resource-conscious and sustainable.

Adaptive Systems: Building AI that automatically adjusts to context and requirements.

Economic Optimization: Balancing performance and cost in AI deployment.

6 Implementation Challenges and Open Questions

6.1 Technical Challenges

Real-time Complexity Assessment: Computing complexity measures quickly enough for interactive applications while maintaining accuracy.

Cold Start Problem: How to perform well on the first few examples when the system has no training data for a new task or domain.

Computational Overhead: Balancing the benefits of adaptive allocation against the computational cost of complexity analysis and strategy selection.

Evaluation Methodology: Developing fair comparisons between adaptive and fixed approaches that account for both performance and cost.

Model Dependency: Understanding how allocation strategies should vary across different base models (GPT-4, Claude, Llama, etc.).

6.2 Open Research Questions

Cross-Modal Extension: How would this approach extend to multimodal inputs (text + images, etc.) where complexity assessment becomes more challenging?

User Personalization: Should the system learn individual user preferences in addition to input characteristics? How can privacy be preserved?

Interactive Adaptation: Could the system adjust allocation based on intermediate model responses, not just initial input analysis?

Meta-Learning: Could a system trained on some tasks quickly adapt to new tasks with minimal additional training data?

Adversarial Robustness: How can the system avoid being gamed by users who might try to exploit complexity measures to get cheaper processing?

7 Economic Impact Analysis

7.1 Quantifying Potential Savings

Based on current pricing models and usage patterns, adaptive allocation could provide substantial economic benefits:

Enterprise Cost Reduction: A large enterprise processing 1 million queries daily with an average prompt length of 500 tokens would spend approximately \$30,000 monthly on input tokens alone with GPT-4 pricing. A 20% reduction through adaptive allocation would save \$6,000 monthly, or \$72,000 annually per application.

API Democratization: More efficient prompting could make advanced capabilities accessible to smaller organizations and developers with limited budgets, expanding the market for AI applications.

Resource Optimization: Reduced token usage could decrease infrastructure requirements for both API providers and users, creating broader economic benefits.

7.2 Performance Value

Beyond cost savings, performance improvements have economic value:

Quality Improvements: Better outputs from adaptive prompting could reduce the need for human review and correction, providing additional cost savings.

Task Completion Rates: More appropriate prompting for complex inputs could improve success rates, reducing retry costs and improving user satisfaction.

8 Call to Action

8.1 What the Research Community Should Do

I believe this research direction deserves investigation by researchers with the computational resources to properly validate these ideas. Specifically, I call for:

Empirical Validation: Large-scale experiments testing the core hypothesis that adaptive allocation improves performance while reducing costs.

Complexity Measure Development: More sophisticated approaches to measuring input complexity across domains, potentially leveraging recent advances in representation learning.

Algorithm Innovation: Novel learning algorithms specifically designed for the allocation optimization problem, incorporating economic constraints and multi-objective optimization.

Benchmark Development: Standardized datasets and evaluation protocols for comparing adaptive prompting approaches, including both performance and cost metrics.

Industry Collaboration: Partnerships with organizations deploying LLMs at scale to validate real-world benefits and constraints.

8.2 Resources and Collaboration

To facilitate this research, I am making available:

- Preliminary framework implementation at `https://github.com/adaptiveprompting/ADAPT-framework`
- Complexity measure implementations and example datasets
- Experimental protocols and baseline implementations
- Economic analysis models and cost-benefit calculators

I welcome collaboration and am particularly interested in connecting with researchers who have the computational resources to conduct large-scale validation studies and access to real-world deployment scenarios.

9 Conclusion

Current prompt engineering approaches use one-size-fits-all token allocation strategies that treat all inputs within a task identically. I argue that this represents a fundamental inefficiency and missed opportunity for both performance improvement and cost reduction.

The ADAPT framework I propose would dynamically allocate prompt tokens based on individual input complexity, potentially leading to improved performance while reducing computational costs. The economic implications are substantial - with current API pricing, even modest efficiency gains could translate to significant cost savings for organizations using LLMs at scale.

While I have outlined the technical approach and identified key research questions, this work requires substantial empirical validation that exceeds the computational resources available to an independent high school researcher. The combination of theoretical promise and practical economic impact makes this a compelling area for investigation.

I believe this research direction has the potential to significantly advance prompt engineering and make AI systems more efficient and effective. The core insight - that optimal prompting should depend on input characteristics, not just task type - appears both theoretically sound and practically important, with clear economic incentives for adoption.

Rather than claiming to have solved this problem, I hope to have made a compelling case that it deserves solution. The transition from task-level to input-level optimization represents a natural next step in the evolution of prompt engineering, and current economic pressures make now the right time to explore these possibilities.

The research community has an opportunity to advance both scientific understanding and practical efficiency simultaneously - a combination that could accelerate the beneficial deployment of AI systems across society.

Acknowledgments

I would like to thank my family for supporting my independent research exploration and the online AI research community for sharing knowledge that helped me understand this field. I'm grateful for access to educational resources and pricing information from OpenAI that made this theoretical and economic analysis possible.

AI Tool Disclosure: This research proposal was developed with significant assistance from AI tools (ChatGPT, Claude AI) for concept refinement, methodology design, literature review, and technical guidance. While the core insight about input-adaptive prompting was my original observation, the framework development, complexity measurement approaches, experimental design, and economic analysis were developed collaboratively with AI assistance. All cited sources were independently verified through web searches and official documentation.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Hajishirzi, et al. Dspy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*, 2023.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, 2020.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837, 2022.