# The Few-shot Dilemma: Over-prompting Large Language Models in Resource-Constrained Environments

**Amine Mike El Maalouf**
amine.el-maalouf@epita.fr

**Cedric Damais**
cedric.damais@epita.fr

**Yacine Benihaddadene**
yacine.benihaddadene@epita.fr

**Leon Ayral**
leon.ayral@epita.fr

## Abstract

Large Language Models (LLMs) have revolutionized NLP through In-Context Learning (ICL). However, a common misconception is "the more examples, the better." Recent research suggests a "Few-shot Dilemma": increasing the number of shots can paradoxically lead to diminishing returns or performance degradation. [3]. Our project investigates this phenomenon specifically within the realm of Small Language Models (SLMs), such as Llama 3.2 and Phi-3. Unlike massive models, SLMs are constrained by smaller context windows. We aim to evaluate various example selection strategies —ranging from semantic retrieval to lexical matching— and systematically vary the number of examples ($K$) to determine the optimal context volume. Our goal is to determine if "smarter" example selection strategies can delay the onset of the few-shot dilemma compared to random selection, and identify the threshold where performance degradation begins, optimizing the trade-off between accuracy and inference cost.

## 1 Introduction: The Real-life Use Case

Consider a privacy-focused Fintech startup building an on-premise "Intent Classification" system to route customer support tickets. Due to data privacy laws (GDPR) and inference costs, they cannot send data to OpenAI/GPT-4; they must run a small model (like Llama 3.2 3B) locally on limited hardware. The engineering team faces a critical challenge: **How do we prompt this small model to get the best accuracy?**

- If they provide 0 examples, accuracy is too low.
- If they provide 50 examples, they blow up the latency and context window, and the model might get confused ("lost in the middle" [2]).

Which examples should they pick from their database of 10,000 tickets? The core challenge is solving the optimization problem between Accuracy, Inference Cost (token usage), and Selection Complexity. We aim to find a "sweet spot" —the **Pareto frontier**— where an SLM achieves near-supervised performance using the minimal number of optimally selected examples.

## 2 Background

Traditionally, NLP relied on Supervised Fine-Tuning (SFT). While SFT yields State-of-the-Art (SOTA) performance, it requires significant compute. The arrival of GPT-3 introduced ICL, allowing models to generalize from just $K$ examples.

Preprint. Under review.

However, in the prompt engineering domain, standard scaling laws are contested.

- **Prompt Sensitivity:** Research shows LLMs are highly sensitive to the order of examples [1].
- **Lost in the Middle:** Liu et al. demonstrated that models often ignore information in the middle of long contexts, focusing only on the beginning and end. [2].
- **The Over-prompting Dilemma:** Adding examples beyond a threshold introduces noise that confuses the attention mechanism.[3]

**Differentiation:** Most benchmarks focus on massive models. We strictly test SLMs ($<$ 4B parameters) and contrast naive Random Sampling against Retrieval Augmented Generation (RAG) techniques (K-NN via Embeddings and TF-IDF).

## 3 Methodology and Project Steps

We implement our pipeline using Python, HuggingFace Transformers, and Scikit-learn.

### 3.1 Phase 1: Infrastructure & Baselines

We utilize datasets like Banking77 (Intent) or CoNLL (NER).

- **Supervised Baseline (Ceiling):** Fine-tuned DistilBERT on the full training set.
- **Zero-Shot Baseline (Floor):** Target LLMs with simple instructions ($K = 0$).

### 3.2 Phase 2: Selector Implementation

We implement the following selection strategies:

- **Random:** `random.sample()` from the train set.
- **Semantic:** Encode queries using `all-MiniLM-L6-v2` to retrieve top-$K$ neighbors.
- **Lexical:** TF-IDF or BM25 to find keyword-similar examples.

We will explore other selection methods, among them cross-encoders.

### 3.3 Phase 3: The Selector Experiment

In this phase, we isolate the impact of the example selection mechanism. To ensure consistency, we fix the model architecture to a static baseline (e.g., `Phi-3-mini`) and vary only the retrieval strategy. We evaluate three primary approaches:

- **Random Sampling:** The naive baseline.
- **Semantic Retrieval:** Using dense embeddings to find contextually similar examples.
- **Lexical Retrieval:** Using TF-IDF or BM25 to find keyword overlap.

We log performance across four key metrics: Accuracy, F1-Score, Inference Latency, and Total Token Count.

### 3.4 Phase 4: The Model Experiment

Building on the results from Phase 3, we fix the selection strategy to the highest-performing method and treat the model architecture as the independent variable. We benchmark a diverse set of open-weights Small Language Models (SLMs) to compare their in-context learning capabilities. The candidate models include `Phi-3-mini`, `Llama-3.2` (1B/3B), `Mistral`, and `Qwen` (specifically for mathematical reasoning tasks). The objective is to identify which architecture maximizes the score for a given optimized prompt structure.

### 3.5 Phase 5: The K-Shot Scaling Experiment

This final experiment conducts a sensitivity analysis by isolating the number of examples ($K$) as the sole independent variable. We will fix the optimal model (from Phase 4) and the most effective selection strategy (from Phase 3) to strictly evaluate the impact of context volume.

**Objective:** We will incrementally increase the number of retrieved examples (e.g., $K \in \{1, 3, 5, 10, 15, 20, \ldots\}$) until the context window is filled. The goal is to empirically locate the "over-prompting" threshold and identify the **sweet spot**—the optimal value of $K$ where the model achieves maximum predictive performance before accuracy plateaus or degrades due to noise.

### 3.6 Phase 6: Evaluation and Analysis

The final phase focuses on synthesizing the experimental data to derive actionable insights regarding the efficiency of SLMs. We will conduct a multi-dimensional analysis to solve the optimization problem defined in our objective.

**Data Visualization** We will generate heatmaps illustrating the relationship between **Accuracy** and **Shot Count** ($K$) for each model architecture. These visualizations will aim to visually highlight the "saturation regions" where adding examples no longer yields performance gains.

**Cost vs. Performance Analysis** To address the constraints of the real-world Fintech use case, we will map the **Pareto frontier**. By plotting Model Accuracy ($y$-axis) against Context Length and Inference Cost ($x$-axis), we aim to identify the optimal operating point that maximizes performance per token.

**Threshold Determination** Finally, we will tabulate the specific "Over-prompting" threshold for each model size. This analysis will determine the exact point where the marginal utility of adding an example becomes negative, while explicitly accounting for the hard constraints of the SLM's context window (e.g., 2048 or 4096 tokens).

## References

[1] Bryan Guan, Tanya Roosta, Peyman Passban, and Mehdi Rezagholizadeh. The order effect: Sensitivity in large language models. *arXiv preprint arXiv:2502.04134*, 2025. URL `https://arxiv.org/html/2502.04134v2`.

[2] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023. URL `https://arxiv.org/abs/2307.03172`.

[3] Yongjian Tang, Doruk Tuncel, Christian Koerner, and Thomas Runkler. The few-shot dilemma: Over-prompting large language models. *arXiv preprint arXiv:2509.13196*, 2025. URL `https://arxiv.org/pdf/2509.13196`.