

How Stable Are Few-Shot Prompting Trade-Offs?

A Cross-Task Cost-Utility Analysis for LLM Text Classification

Giorgos Kyriakou
Technical Computer Science
University of Twente

June 26, 2026

Abstract

Few-shot prompting is commonly presented as a lightweight way to adapt large language models to downstream tasks without training a new model. In practice, however, every example added to a prompt increases input length and token cost, while latency is also relevant as a deployment metric. This creates a practical engineering question: when do examples improve classification enough to justify their extra resource use? This paper reports an empirical study of this question across four text classification tasks: TREC question type classification, Emotion classification, News headline topic classification, and Twitter financial sentiment classification. For each task, `openai/gpt-4o-mini` was evaluated through OpenRouter using one prompt framework, five fixed example pools, five shot levels, and 500 test examples per pool-shot condition. The final reported experiment contains 50,000 model calls. Under the zero-shot-relative evaluation rule used in this study, zero-shot emerged as the safest overall baseline recommendation across the reported tasks. In quality terms, TREC, Emotion, and Twitter also favored zero-shot on aggregate, while News achieved its highest mean accuracy and macro-F1 at 10-shot. The main contribution is therefore not a universal recommendation to avoid few-shot prompting, but a framework for measuring whether a few-shot prompt is worth its cost across tasks and example pools.

Keywords: large language models, prompt engineering, few-shot prompting, in-context learning, text classification, cost-utility analysis, token cost, prompt stability

1 Introduction

Large language models (LLMs) can perform many language tasks from natural-language instructions and examples without updating model parameters [1]. This has made prompt-based text classification a serious alternative to task-specific fine-tuning in settings with limited annotated data [9]. In practical API-based workflows, prompting can also be at-

tractive when retraining a separate classifier is inconvenient. Instead of fitting a separate classifier, a developer can describe the task in natural language and ask a general-purpose model to return one label from a fixed set.

One common way to adapt prompt-based behavior is few-shot prompting, where labelled examples are included before the target input [1]. Demonstrations can communicate the label space, input distribution, and expected output format, not only the literal input-output mapping [3]. However, the engineering trade-off is less simple. Every added example increases input length and token cost. In API-based LLM use, these resource costs can become substantial at scale [15]. Longer prompts may also affect latency, although request time can vary with network, provider, and runtime conditions. The practical question is not whether few-shot prompting can help in principle, but whether it helps enough, consistently enough, and cheaply enough for a specific task.

The literature gives reasons to be cautious about assuming that more examples are automatically better. Example choice and example order can substantially affect in-context learning performance [4, 5]. Active-selection and demonstration-count studies further suggest that the value of additional examples can vary by task, selected examples, and marginal shot-count transition [6, 7]. Prompt-based classification is also sensitive to the way the task is expressed. Noisy-channel prompting work shows that alternative scoring formulations can improve robustness to prompt and verbalizer variation [8]. Studies of LLM text classification show that prompt complexity and label design can affect classification performance across tasks and domains [9, 11]. Work on many-label classification further shows that label-set size and label semantics can change the difficulty of in-context classification [10]. Budget-aware ICL work also frames limited demonstration or annotation budget as a practical constraint, even in settings beyond sentence-level classification [14]. The gap, then, is not whether examples can matter, but whether a prompt engineer can rely on a stable shot-count recommendation once quality, cost, and pool variation are considered together.

The literature-gap matrix in Table 1 clarifies the intended contribution. It is used as a positioning matrix rather than as a claim that earlier papers are unrelated to these dimensions. Prior work motivates in-context learning, example sensitivity, and LLM classification, but it does not directly answer the practical question addressed here. The distinct focus of this thesis is resource-aware shot-count evaluation: whether a prompt engineer can trust a shot-count recommendation

TScIT 45, July 3, 2026, Enschede, The Netherlands

© 2026 University of Twente, Faculty of Electrical Engineering, Mathematics and Computer Science.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Table 1: Positioning of this study relative to prior work. The table highlights the combination of features that this thesis brings together. Entries indicate whether a dimension is a central evaluated focus of the cited study in the sense used here, not whether it is ever mentioned. Cost refers to the balance between cost and quality, and stability refers to the stability of a practical shot-count recommendation across tasks or example pools.

Study	Few-shot prompting	Example sensitivity	Text classification	Cost and quality balance	Recommendation stability
Brown et al. 2020 [1]	yes	no	mixed	no	no
Min et al. 2022 [3]	yes	yes	yes	no	no
Liu et al. 2022 [4]	yes	yes	mixed	no	no
Lu et al. 2022 [5]	yes	yes	yes	no	no
Zhang et al. 2022 [6]	yes	yes	yes	no	no
Chen et al. 2023 [7]	yes	limited	mixed	limited	no
Cheng et al. 2023 [15]	yes	no	mixed	yes	no
Mu et al. 2024 [11]	limited	no	yes	reported, not focus	no
Edwards and Camacho-Collados 2024 [9]	yes	limited	yes	no	no
Li and Flanigan 2024 [17]	yes	no	yes	no	no
Berger et al. 2025 [14]	yes	yes	no	yes	no
This study	yes	yes	yes	yes	yes

once token cost and pool variation are included in the evaluation.

Research question. For text classification with a fixed LLM and prompting framework, does a stable most cost-efficient shot-count recommendation emerge across task types and example pools when few-shot conditions are evaluated relative to the zero-shot baseline?

Sub-questions.

1. How do accuracy, macro-F1, latency, token usage, and cost vary across 0, 1, 3, 5, and 10 shots?
2. Where, if anywhere, do marginal quality gains justify the added resource cost of more examples?
3. How much does the best shot level vary across fixed example pools?
4. What shot-count recommendation is most defensible for prompt engineering practice under this setup?

To answer these questions, this thesis uses a controlled empirical experiment with one fixed model, one fixed prompting framework, and fixed shot levels. Four classification tasks are used to broaden the comparison: TREC question type classification, Emotion classification, News headline topic classification, and Twitter financial sentiment classification. For each task, the experiment evaluates 0, 1, 3, 5, and 10 shots with five independent example pools. This design treats prompt construction as an engineering decision rather than a single benchmark score: a few-shot setting must be judged by its quality gain, added resource cost, and stability across plausible example pools.

2 Background and Related Work

Prompt-based classification uses a language model as a conditional text generator: the model receives a task description and input text, then produces a label or label-like output [9, 11]. In zero-shot prompting, the model receives only the instruction and label descriptions. In few-shot prompting, the model also receives labelled examples before the target input [1]. These examples do not update model parameters. They condition the model through the context window, which is why the method is often described as in-context learning rather than training [1, 3].

In the setup used here, each prompt has several functional parts. The instruction defines the task, the label list defines the allowed output space, the demonstrations provide optional examples of the mapping, and the final query is the item to be classified. In a classification setting, controlling these parts matters because prompt complexity and label design can affect model behavior [9, 11]. A prompt that allows free-form explanations may produce useful natural language but can make automatic evaluation harder. For this reason, the prompting setup in this thesis constrains the output to one label from a fixed label set.

This makes zero-shot and few-shot prompting useful comparison settings for this study. Zero-shot provides the instruction-only baseline: it tests how well the model can infer the intended mapping from the task description and label set alone. Few-shot prompting tests whether demonstrations improve that behavior by reducing ambiguity and showing concrete label usage. Comparing the two separates what the model can already do from what extra prompt examples add under the fixed prompt format.

This distinction also separates prompting from parameter adaptation. In fine-tuning, model weights are updated using labelled data, while in-context learning keeps the model fixed and supplies examples only at inference time [1, 16]. This makes few-shot prompting attractive when rapid deployment or changing label sets are important, but it also means the same demonstration examples must be paid for repeatedly in every API call. The resource cost of examples is therefore part of the method itself, not just an implementation detail [15].

The theoretical appeal of in-context learning is that demonstrations can reduce ambiguity. A task instruction such as “classify this text” may be interpreted in several ways, but examples can show how labels should be applied in practice. For classification, demonstrations may also communicate output format, label space, and aspects of the input distribution [3]. At the same time, demonstrations are not free. In an API setting, every added example consumes input tokens and raises monetary cost [15]. Latency is also relevant in deployment, but it can vary with provider and runtime effects rather than following prompt length as cleanly as token usage does. If the model already understands the task and the label space well enough from the instruction alone, the extra examples may function mainly as overhead. The eval-

uation must therefore consider both predictive quality and resource use.

Prompt wording is another source of variation. Labels are not only class identifiers. They are also pieces of language that the model can interpret. Descriptive labels may make a task easier in zero-shot settings, while ambiguous or unfamiliar labels may make demonstrations more useful. Prior work on LLM text classification has shown that prompt complexity and label design can affect performance [9, 11]. This background is important for the present study because shot count is evaluated under one fixed prompt format, so the conclusions are about the behavior of that controlled setup rather than about all possible prompt designs.

Example selection is especially important. Liu et al. show that the choice of in-context examples can affect performance [4], and Lu et al. show that the order of examples can also matter [5]. Zhang et al. study active example selection, which further suggests that not all examples are equally useful [6]. For a deployed prompt, this creates a stability issue. If 5-shot prompting works well for one example pool but not another, then the shot-count recommendation is fragile.

This motivates the use of both predictive and resource-oriented evaluation criteria. Accuracy shows the overall fraction of correct predictions, while macro-F1 helps reveal whether performance is balanced across labels rather than dominated by common classes. In an API-based workflow, however, quality alone is not enough. Token usage affects monetary cost, and latency affects the practical deployment profile even when it is noisier than token-based measures [15]. A shot-count recommendation is therefore only useful in practice if it reflects both classification quality and the resources required to obtain it.

3 Methodology

This methodology evaluates prompting as a cost-utility problem. In this study, utility refers to predictive quality, measured with accuracy and macro-F1. Cost refers to the resources recorded for each API call: input and output tokens, monetary API cost, and measured client-side latency. The unit of comparison is not a single response in isolation, but a pool-shot condition: one task, one example pool, one shot level, and 500 test examples.

In this study, “cost-efficient” is treated as a zero-shot-relative dominance judgment. This does not treat zero-shot as universally preferable. Instead, zero-shot is the cheapest prompt condition and functions as the baseline that few-shot prompting must improve upon to justify added resources. A few-shot condition is considered more cost-efficient only when its observed accuracy or macro-F1 gain is large enough to justify higher token and monetary cost. Latency is included as a resource metric, but this study does not define a fixed latency cutoff because acceptable latency depends on the deployment context. The analysis therefore reports observed quality gain, added resource cost, latency, and pool stability so that the trade-off is explicit.

The methodology has five stages. First, each dataset is normalized into a common format with an identifier, a text field, and a gold label. Second, fixed example pools are sampled

from training data only. Third, prompt variants are generated for each pool and shot level. Fourth, model calls are executed through OpenRouter while restricting routing to the OpenAI provider. Fifth, a unified metrics pipeline reads the raw JSONL logs and produces the aggregate outputs used for analysis, including run-level metrics, shot-level comparisons, zero-shot comparisons, marginal transitions, pool stability summaries, per-label metrics, and invalid-output counts.

Each prompt uses the same structure within a task: a task instruction, the valid label list, brief label meanings, an optional examples block, and the target input. The zero-shot condition therefore still receives the task instruction, label set, and label meanings, but no labelled demonstrations. This makes zero-shot a realistic prompt-engineering baseline rather than an intentionally under-specified prompt. Few-shot variants use the same template and add examples from the fixed pool before the target input. The prompt also instructs the model to return only one valid label, which supports automatic parsing and fair comparison across conditions.

The integration of these components is important because the research question is about stability rather than one-off performance. A single prompt may give a misleading impression if its examples happen to be unusually helpful or harmful. Repeating the same shot levels across five pools makes it possible to separate aggregate behavior from pool-specific behavior. Similarly, using four task types makes it possible to test whether a result observed on TREC also appears for affective, topic, and sentiment classification.

To preserve comparability across conditions, the same sampling and evaluation procedure is applied to every task. Example pools are constructed from training data only, while evaluation is performed on separate test subsets. This separation ensures that prompt demonstrations do not contain the same rows later used for measurement. Using the same pool construction logic, shot levels, parser rules, and evaluation pipeline across tasks makes the resulting comparisons more reproducible and easier to interpret.

The example pool is therefore treated as part of the experimental variable rather than as incidental prompt content. Five fixed pools are used per task. Lower shot levels are prefixes of the same pool: 1-shot uses the first example, 3-shot uses the first three, 5-shot uses the first five, and 10-shot uses the full pool. This nested design makes marginal transitions interpretable because moving from 3 to 5 shots means adding two examples to the same prompt family rather than switching to a different demonstration set.

To keep the comparison controlled, the model, provider route, decoding settings, parser, shot levels, and evaluation procedure are held constant across runs. The experimental variation comes from the task, the example pool, and the number of demonstrations included in the prompt. This makes it possible to attribute observed differences more directly to shot count and pool variation rather than to unrelated changes in model behavior or system configuration.

The model and decoding settings were held fixed. All runs used `openai/gpt-4o-mini` through OpenRouter, restricted to the OpenAI provider with fallback disabled. This model was chosen as a practical balance of capability, la-

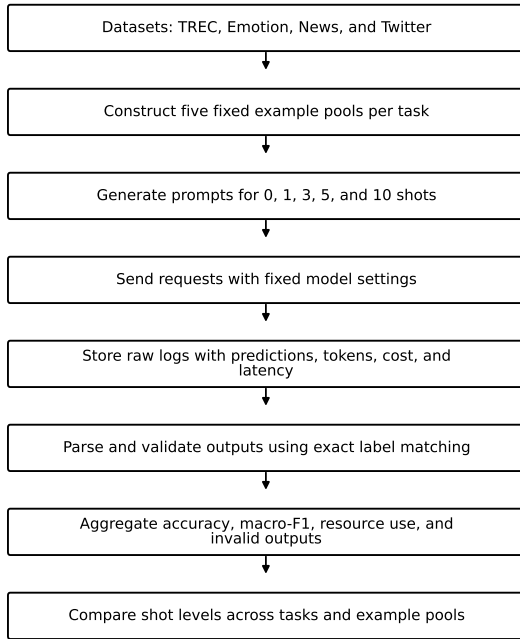


Figure 1: Experiment and analysis pipeline. The same process is repeated for each task, pool, and shot level.

tency, and cost for a large repeated-call experiment. Because the study requires 50,000 API calls across tasks, pools, and shot levels, a lower-cost model was needed to keep the evaluation feasible while still using a modern general-purpose LLM. Keeping the model fixed also isolates the effect of shot count and example-pool variation, rather than conflating those factors with differences between models. Temperature was set to 0, and outputs were capped at 8 tokens. The output cap is sufficient because all valid labels tokenize to short outputs under the model tokenizer. The parser accepts only labels from the fixed label list for the task, and invalid outputs are recorded rather than silently corrected.

4 Experiment Setup

The experiment uses four single-label classification tasks. The Text REtrieval Conference (TREC) question classification dataset is used as a coarse question-classification task with six labels: ABBR, DESC, ENTY, HUM, LOC, and NUM [2]. Emotion uses the public `dair-ai/emotion` release of CARER, the Contextualized Affect Representations for Emotion Recognition dataset, with labels *sadness*, *joy*, *love*, *anger*, *fear*, and *surprise* [18, 19]. News uses a headline topic dataset with six selected labels: Politics and Government, Business and Economy, Sports, Technology and Science, Entertainment and Culture, and Health and Medicine [20]. Twitter uses a financial-news sentiment dataset with three labels: *bearish*, *bullish*, and *neutral* [21].

The initial study design used TREC as a controlled question-classification task. The final experiment extends the same design to three additional single-label classification tasks in order to test whether the observed shot-count pattern is specific to TREC or remains visible across domains.

These tasks were chosen to provide variation in domain and

label semantics while keeping the overall prediction format fixed. TREC represents short factual question classification, Emotion represents affective interpretation, News represents topical categorization, and Twitter represents financial sentiment judgment. Together, they make it possible to test whether the same shot-count recommendation holds across tasks that differ in topic, label meaning, and apparent difficulty, without changing the basic single-label classification setting.

Table 2: Reported task set. Each task uses five pools, five shot levels, and 500 test examples per condition.

Task	Type	Labels	Calls
TREC	Question type	6	12,500
Emotion	Affective	6	12,500
News	Topic	6	12,500
Twitter	Sentiment	3	12,500
Total			50,000

Each task used five fixed example pools. The highest shot condition was 10-shot, so every pool contains ten examples. For six-label tasks, each pool includes all six labels at least once and four repeated labels. For Twitter, the pool contains a near-balanced distribution across *bearish*, *bullish*, and *neutral*. Test subsets were sampled separately from the prompt examples, and no test row appears in the prompt pools.

The tested shot levels were 0, 1, 3, 5, and 10. This range was chosen to cover both a no-demonstration baseline and several practically plausible few-shot settings while keeping the full comparison economically feasible. Including 1-shot makes it possible to observe the immediate effect of adding a single example, while 3-shot and 5-shot represent modest few-shot settings and 10-shot tests a larger demonstration budget. The total reported experiment is:

$$4 \text{ tasks} \times 5 \text{ pools} \times 5 \text{ shot levels} \times 500 \text{ rows} = 50,000 \text{ calls}$$

The evaluation operationalizes the cost-utility definitions from the methodology. Quality is measured with accuracy and macro-F1. Resource use is measured through input tokens, output tokens, total tokens, OpenRouter cost metadata, and latency. Latency is measured client-side as elapsed wall-clock time between sending the OpenRouter API request and receiving the HTTP response, so it captures practical request latency rather than provider-reported model-only generation time. These measures support the zero-shot-relative comparison: a few-shot condition must be compared against the cheapest prompt, not only against other few-shot prompts.

Across the final experiment, only five outputs were invalid under the exact-label parser. All occurred in the Emotion task and were counted as incorrect, preserving the fixed-label evaluation rule.

5 Results and Discussion

5.1 Aggregate results

The aggregate results show the main pattern of the study: zero-shot is the cheapest condition and has the highest aggregate quality for TREC, Emotion, and Twitter, while News

Table 3: Aggregate quality and resource results by shot level. Tokens and cost are averaged per 500-test-example pool-shot condition. Latency is mean client-side request latency in seconds.

Task	Shots	Mean acc.	Macro-F1	Tokens	Latency	Mean cost	Cost x vs 0
TREC	0	0.8556	0.8546	82,904	0.749	0.012720	1.00
TREC	1	0.8048	0.7996	93,413	0.756	0.014301	1.12
TREC	3	0.8160	0.8190	113,514	0.770	0.017316	1.36
TREC	5	0.8204	0.8227	129,305	0.793	0.019681	1.55
TREC	10	0.8288	0.8337	174,315	0.731	0.026437	2.08
Emotion	0	0.5904	0.5105	87,955	0.716	0.013515	1.00
Emotion	1	0.5820	0.4772	103,855	0.766	0.015900	1.18
Emotion	3	0.5872	0.5039	130,127	0.772	0.019828	1.47
Emotion	5	0.5796	0.4974	155,021	0.779	0.023559	1.74
Emotion	10	0.5900	0.4990	229,824	0.781	0.034781	2.57
News	0	0.8808	0.8815	102,436	0.497	0.015967	1.00
News	1	0.8700	0.8693	116,231	0.484	0.018034	1.13
News	3	0.8672	0.8681	142,129	0.512	0.021918	1.37
News	5	0.8732	0.8742	166,031	0.514	0.025504	1.60
News	10	0.8908	0.8912	225,431	0.497	0.034414	2.16
Twitter	0	0.8592	0.8613	67,341	0.496	0.010460	1.00
Twitter	1	0.8364	0.8375	83,460	0.529	0.012887	1.23
Twitter	3	0.8428	0.8440	107,654	0.520	0.016513	1.58
Twitter	5	0.8448	0.8457	136,759	0.505	0.020881	2.00
Twitter	10	0.8456	0.8461	215,467	0.534	0.032691	3.13

is the main quality-side exception. Table 3 should therefore be read through three columns together: mean accuracy, macro-F1, and cost relative to zero-shot. For TREC, Emotion, and Twitter, adding examples increases cost without improving the aggregate quality result. TREC is the clearest case: 10-shot costs 2.08 times as much as zero-shot and still has lower accuracy and lower macro-F1. Twitter shows a similar pattern, with 10-shot costing 3.13 times as much as zero-shot and reducing mean accuracy by 1.36 percentage points.

News changes the story in quality terms. For News, 10-shot improves mean accuracy from 0.8808 to 0.8908 and macro-F1 from 0.8815 to 0.8912. This is an observed aggregate few-shot gain, but it is not free: the mean cost rises from 0.015967 to 0.034414 USD per 500-row condition. The gain is therefore approximately one percentage point of accuracy for a little more than double the cost. The News result is important because it shows that the zero-shot recommendation is not a general rejection of few-shot prompting. In tasks where examples may help clarify a task-specific label space, few-shot prompting can still be beneficial. Instead, the results support a more defensible conclusion: few-shot prompting can improve quality for some tasks, but whether it is worth adopting depends on whether that gain is large enough to justify its token cost.

The zero-shot-relative comparison makes this pattern clearer. Every task shows a negative aggregate accuracy change at 1-shot relative to zero-shot, so adding a single example is not a reliable low-cost improvement in this setup. News is the only task with a positive aggregate 10-shot delta, while TREC shows the largest early drop, with accuracy falling by 5.08 percentage points from 0-shot to 1-shot.

The aggregate comparison is important because it shows which pattern survives after averaging across all five pools for a task. On both mean accuracy and macro-F1, zero-shot is the best aggregate condition for TREC, Emotion, and Twitter, while 10-shot is the best aggregate condition for News. This makes the result more nuanced than a simple zero-shot win: some few-shot gains are local, and News shows a stable quality-side benefit, but that benefit still comes with substantially higher token cost.

This pattern is consistent with prior work showing that

demonstrations do not necessarily improve in-context performance through stronger task discrimination alone, and may instead help by regulating label space and output format [3, 12].

5.2 Shot-count trajectories

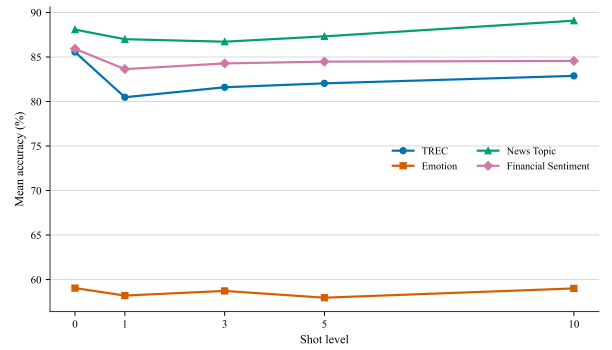


Figure 2: Mean accuracy by shot level across the four reported tasks.

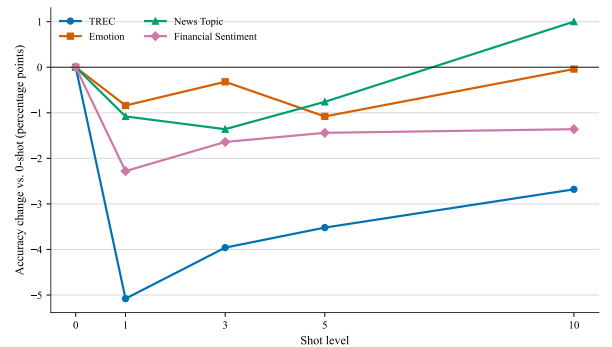


Figure 3: Accuracy change relative to zero-shot. News is the only task with a positive aggregate 10-shot gain.

The shot-count trajectories show that adding examples does not produce monotonic quality gains. Figures 2 and 3 show why the result should be described as task-dependent: TREC and Twitter remain below the zero-shot baseline across later shot levels, while Emotion fluctuates and ends close to zero-shot in accuracy but below it in macro-F1. News is different:

after an early decline, later shot levels recover and 10-shot becomes the highest aggregate quality condition. This suggests that examples may be useful when the label categories are broader, more context-dependent, or less directly inferable from label names alone.

5.3 Cost and pool stability

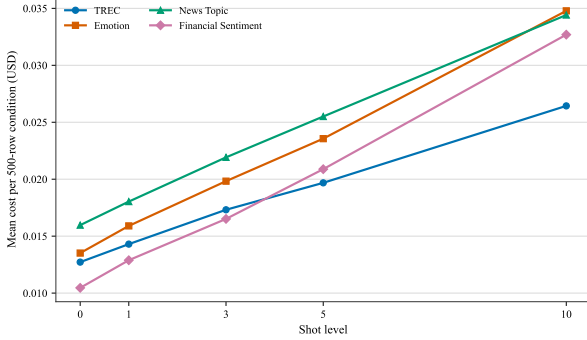


Figure 4: Mean cost rises as shot level increases. The cost trend is consistent even when quality trends differ.

The resource side is more predictable than the quality side. Figure 4 shows that token usage and monetary cost rise consistently because each added demonstration lengthens the prompt. Client-side latency is reported in Table 3, but interpreted more cautiously because it can vary with network, provider, and runtime conditions. Few-shot conditions therefore require a clear quality gain.

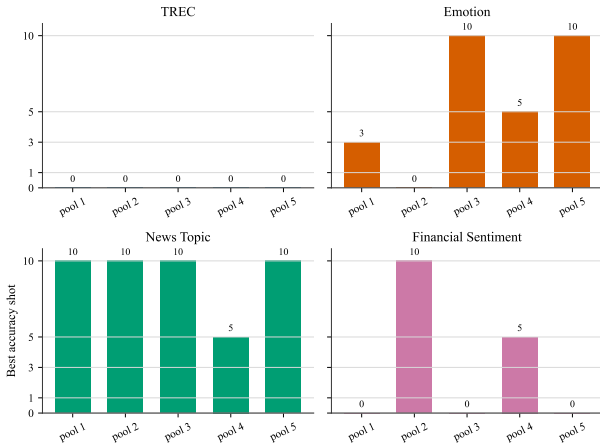


Figure 5: Best accuracy shot by pool. The best shot is stable for TREC, variable for Emotion and Twitter, and mostly 10-shot for News.

The pool-level results show why stability matters for prompt engineering decisions. TREC is stable: zero-shot is best in every pool. News is also fairly stable, but in the opposite direction: few-shot beats zero-shot in every pool, and 10-shot is best in four out of five pools. Emotion and Twitter are less stable. Emotion has local few-shot wins in four out of five pools, even though zero-shot remains best on aggregate. Twitter has few-shot wins in two out of five pools, but zero-shot remains the aggregate winner. A prompt engineer who tests only one example pool could therefore choose a different shot level from the aggregate recommendation. This is why the example pool is part of the evaluation rather than incidental prompt content.

This variation is consistent with prior work showing that the choice of demonstrations can change in-context performance [4, 6]. It also matches work on prompt-order sensitivity, where the same examples can behave differently depending on their arrangement [5].

The marginal-transition results show the marginal quality side of the cost-utility trade-off. In the aggregate metrics, the first step from 0-shot to 1-shot reduces accuracy for all four tasks, while the added example also increases token use and monetary cost. For example, averaged across the five pools, the 0-shot to 1-shot transition reduces accuracy by 5.08 percentage points for TREC and by 1.08 percentage points for News, while still increasing cost in both cases. Later transitions sometimes recover part of that quality loss, but usually not enough to overturn the zero-shot baseline. News is the only task where later few-shot gains are large enough to produce the highest aggregate quality condition. Averaged across the five pools, its 5-shot to 10-shot transition increases accuracy by 1.76 percentage points, but also adds 0.00891 USD per 500-test-example condition. Even there, the gain must be judged against the cumulative cost increase. This explains why few-shot prompting should be evaluated as a staged trade-off rather than as a single yes-or-no intervention.

5.4 Task-wise interpretation

The TREC results suggest that few-shot examples may be least useful when the label space is already direct and well described. The labels correspond to broad question-answer types, and the prompt gives short explanations of each label. Under this setup, the model may already infer much of the intended mapping from the instruction and label meanings alone. Adding examples increases cost but does not improve the aggregate result in this experiment.

Emotion behaves differently, and one plausible reason is that the labels are semantically closer to one another. Categories such as sadness, fear, anger, and love can overlap in natural language, and a short text may contain signals for more than one affective state. This may help explain why some pools produce local few-shot wins while the aggregate result remains close to zero-shot. The gap between accuracy and macro-F1 also matters here. It suggests that the model may classify some emotion labels better than others, so overall accuracy does not fully describe performance across the label set.

News is the clearest quality-side exception. The topic labels are broad, and headlines can mention people, companies, events, or institutions in ways that cross category boundaries. One plausible interpretation is that examples helped the model align with how this particular dataset uses those categories, which is consistent with later few-shot levels recovering from the early drop and 10-shot becoming the highest aggregate quality condition. However, the cost-efficiency decision is still not automatic, because that quality gain comes with a higher token and monetary cost.

Twitter shows that a small label set does not guarantee that few-shot examples will help. Although the task has only three labels, financial sentiment can depend on domain cues such as upgrades, downgrades, earnings, liquidity, and mar-

ket expectations. The prompt already describes the sentiment labels with financial meanings, so examples may add less new information than expected. Pool-level variation is also consistent with examples sometimes being unhelpful rather than producing a stable improvement.

5.5 Answering the research questions

The results answer SQ1 by showing that token usage and monetary cost increase consistently with shot count, while quality does not increase consistently. Client-side latency is noisier but remains part of the reported resource profile. They answer SQ2 by showing that marginal quality gains are not guaranteed and often do not justify marginal cost increases: in the aggregate metrics, the first transition from 0 to 1 shot is negative for all four tasks, and later gains often do not recover the zero-shot baseline. They answer SQ3 by showing that pool sensitivity exists and differs by task. They answer SQ4 by supporting zero-shot as the safest default starting point in this setup, not as a universal final choice. In a practical workflow, few-shot prompting should be adopted only when its task-specific gain is large enough to justify added cost.

6 Conclusion

This thesis studied whether few-shot prompt size has a stable cost-efficient recommendation across text classification tasks. Under the zero-shot-relative cost-efficiency rule used here, zero-shot is the safest default in this setup, but the quality-side results are task-dependent. Across 50,000 model calls, zero-shot had the highest aggregate mean accuracy while also remaining the cheapest condition for TREC, Emotion, and Twitter. News was the main exception, achieving its highest aggregate accuracy and macro-F1 at 10-shot. Pool-level results also show that some few-shot conditions can win locally even when zero-shot remains the aggregate recommendation. The main conclusion is therefore not that few-shot prompting is unnecessary, but that zero-shot should be treated as a serious baseline and few-shot prompting should be justified empirically rather than assumed to improve performance.

The project has several limitations. It uses one model, one provider route, one prompt template, and four classification tasks. The study is scoped as a descriptive cost-utility evaluation, so small differences are treated as observed patterns rather than statistically confirmed effects. The example pools are fixed and reproducible, but not optimized or retrieval-based, so the study does not test whether smarter example selection would change the preferred shot count. Some datasets are public and older, so benchmark or task contamination cannot be ruled out for a closed model [17]. The News dataset is newer, but it is an applied dataset rather than a long-established academic benchmark. In addition, the results are limited to single-label classification and to one API-based deployment setting. The study also does not test sensitivity to prompt rephrasings, even though prompt wording itself can affect robustness and consistency [13]. Finally, it does not compare in-context prompting against few-shot fine-tuning or smaller alternative models, even though prior

work shows that such comparisons can change the interpretation of adaptation performance [16].

Future work could extend this study by testing additional models, prompt templates, and example-selection strategies. Retrieval-based, optimized, or order-aware demonstration strategies would help determine whether the limited and task-dependent few-shot gains observed here are due to shot count itself or to the use of fixed, non-optimized pools [4, 5, 6]. Repeating the analysis with full test sets, more task types, and uncertainty estimates would make it easier to distinguish small observed differences from robust effects. A broader extension would compare in-context prompting with fine-tuning or smaller task-specific models, placing the cost-utility results in a wider deployment context [16].

The practical implication is straightforward. In LLM applications, adding examples is a resource decision as well as a modeling decision. A developer should first measure a strong zero-shot prompt, then test few-shot prompts across multiple example pools before accepting the added cost. Few-shot prompting should still be tested when the task appears likely to benefit from examples, but it should be adopted only when its task-specific gain is large enough to justify the added resources. The dashboard and metric pipeline built for this project support that workflow by enabling run control, prompt inspection, raw-response logging, and reproducible comparison of the reported quality, cost, latency, and stability metrics.

AI acknowledgement. During the preparation of this thesis, I used artificial intelligence tools as a limited aid for writing and coding. In the written thesis, AI was used mainly to revise wording, improve clarity, and check grammar in text that I had written. For the coding component, AI was used for programming and implementation support, including checking syntax, explaining error messages, suggesting possible fixes during debugging, and helping refine the layout and visual presentation of the dashboard frontend. All AI-assisted text and code suggestions were reviewed, edited, adapted, and tested by me where relevant before inclusion. The research design, experiments, analysis, interpretation of results, implementation choices, and final thesis content remain my own responsibility.

References

- [1] T. B. Brown et al. Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems*, 2020.
- [2] X. Li and D. Roth. Learning Question Classifiers. In *Proceedings of COLING*, 2002.
- [3] 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? In *Proceedings of EMNLP*, 2022.
- [4] J. Liu, D. Shen, Y. Zhang, B. Dolan, L. Carin, and W. Chen. What Makes Good In-Context Examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO)*, 2022.

- [5] Y. Lu, M. Bartolo, A. Moore, S. Riedel, and P. Stenetorp. Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity. In *Proceedings of ACL*, 2022.
- [6] Y. Zhang, S. Feng, and C. Tan. Active Example Selection for In-Context Learning. In *Proceedings of EMNLP*, 2022.
- [7] J. Chen, L. Chen, C. Zhu, and T. Zhou. How Many Demonstrations Do You Need for In-Context Learning? In *Findings of EMNLP*, 2023.
- [8] S. Min, M. Lewis, H. Hajishirzi, and L. Zettlemoyer. Noisy Channel Language Model Prompting for Few-Shot Text Classification. In *Proceedings of ACL*, pages 5316–5330, 2022.
- [9] A. Edwards and J. Camacho-Collados. Language Models for Text Classification: Is In-Context Learning Enough? In *Proceedings of LREC-COLING*, pages 10058–10072, 2024.
- [10] A. Milios, S. Reddy, and D. Bahdanau. In-Context Learning for Text Classification with Many Labels. In *Proceedings of the 1st GenBench Workshop on (Benchmarking) Generalisation in NLP*, pages 173–184, 2023.
- [11] Y. Mu, B. P. Wu, W. Thorne, A. Robinson, N. Aletras, C. Scarton, K. Bontcheva, and X. Song. Navigating Prompt Complexity for Zero-Shot Classification: A Study of Large Language Models in Computational Social Science. In *Proceedings of LREC-COLING*, 2024.
- [12] Q. Long, Y. Wu, W. Wang, and S. J. Pan. Does In-Context Learning Really Learn? Rethinking How Large Language Models Respond and Solve Tasks via In-Context Learning. In *Proceedings of COLM*, 2024.
- [13] F. Errica, G. Siracusano, D. Sanvito, and R. Bifulco. What Did I Do Wrong? Quantifying LLMs’ Sensitivity and Consistency to Prompt Engineering. In *Proceedings of NAACL*, 2025.
- [14] U. Berger, T. Baumel, and G. Stanovsky. In-Context Learning on a Budget: A Case Study in Token Classification. In *Proceedings of Insights*, 2025.
- [15] Z. Cheng, J. Kasai, and T. Yu. Batch Prompting: Efficient Inference with Large Language Model APIs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, 2023.
- [16] M. Mosbach, T. Pimentel, S. Ravfogel, D. Klakow, and Y. Elazar. Few-shot Fine-tuning vs. In-context Learning: A Fair Comparison and Evaluation. In *Findings of ACL*, 2023.
- [17] C. Li and J. Flanigan. Task Contamination: Language Models May Not Be Few-Shot Anymore. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.
- [18] E. Saravia, H.-C. T. Liu, Y.-H. Huang, J. Wu, and Y.-S. Chen. CARER: Contextualized affect representations for emotion recognition. In *Proceedings of EMNLP*, pages 3687–3697, 2018.
- [19] DAIR.AI. Emotion. Hugging Face dataset card, 2025. <https://huggingface.co/datasets/dair-ai/emotion>.
- [20] Momentum Lab. News Categories. Hugging Face dataset card, 2025. <https://huggingface.co/datasets/momentum-lab/news-categories>.
- [21] Zeroshot. Twitter Financial News Sentiment. Hugging Face dataset card, 2022. <https://huggingface.co/datasets/zeroshot/twitter-financial-news-sentiment>.