SpareLLM: Automatically Selecting Task-Specific Minimum-Cost Large Language Models under Equivalence Constraint

SAEHAN JO, Cornell University, USA
IMMANUEL TRUMMER, Cornell University, USA

We introduce SpareLLM, **S**electing **P**assable **A**nd **R**esource-**E**fficient **LLM**s, a novel LLM framework designed to minimize the inference costs (i.e., resource-efficient) of large-scale NLP tasks while ensuring sufficient result quality (i.e., passable). It enables users to specify an equivalence constraint in terms of the equivalence of outputs to those of the most powerful LLM. SpareLLM then generates results that deviate from the outputs of this LLM only with a probability below a user-defined threshold. SpareLLM employs a profiling phase that evaluates the performance of multiple LLMs to identify those that meet the user-defined equivalence level. It optimizes the tradeoff between profiling overheads and the anticipated cost savings resulting from profiling. Moreover, SpareLLM further reduces inference costs by strategically leveraging a mix of LLMs. Our experiments on five real-world datasets show that SpareLLM achieves significant cost savings, up to 8.6×, while generating equivalent outputs in 90% of cases compared to GPT-4-Turbo. Compared to recent LLM cascading baselines, SpareLLM demonstrates a superior tradeoff between cost and accuracy, accounting for 91.1% and 83.8% of the points on the Pareto curve for OpenAI and Llama models.

CCS Concepts: • **Computing methodologies** → **Natural language processing**; **Machine learning**; • **Information systems**;

Additional Key Words and Phrases: large language models, approximate processing, natural language processing

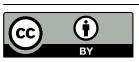
ACM Reference Format:
Saehan Jo and Immanuel Trummer. 2025. SpareLLM: Automatically Selecting Task-Specific Minimum-Cost Large Language Models under Equivalence Constraint. *Proc. ACM Manag. Data* 3, 3 (SIGMOD), Article 219 (June 2025), 26 pages. https://doi.org/10.1145/3725356

1 Introduction

The rapid advancement and deployment of Large Language Models (LLMs) have started a new era of artificial intelligence (AI). The capabilities of AI systems have significantly improved across a variety of natural language processing (NLP) tasks. Companies such as OpenAI [30] and Anthropic [32] have been at the forefront, offering LLMs as services to enable users to build intelligent, automated solutions. However, deploying these models comes with substantial costs, especially for large-scale inference. In recent years, there has been a notable trend in LLM development toward increasing the number of parameters to boost performance [15, 20]. The expansion in model size has directly contributed to higher inference costs, making it more expensive for end-users as well.

Service providers often offer a variety of LLM options. For instance, OpenAI provides models such as GPT-4, GPT-3.5, Davanci, and Babbage [30]. Similarly, Google offers four sizes of its Gemini model: Ultra, Pro, Flash, and Nano [13]. These LLMs exhibit heterogeneous performance and costs,

Authors' Contact Information: Saehan Jo, Cornell University, Ithaca, New York, USA, sj683@cornell.edu; Immanuel Trummer, Cornell University, Ithaca, New York, USA, itrummer@cornell.edu.



ACM 2836-6573/2025/6-ART219
https://doi.org/10.1145/3725356

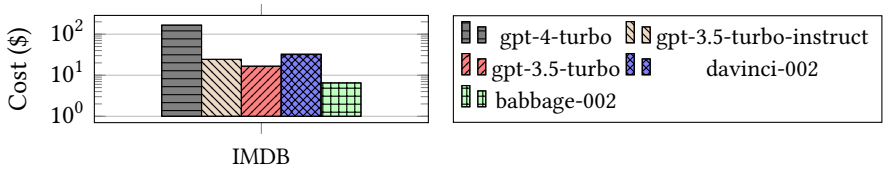


Fig. 1. Costs of OpenAI LLMs for the sentiment classification task on the IMDB benchmark.

in which a more powerful LLM is inevitably more expensive. The cost difference between these LLMs can span several orders of magnitude, as illustrated in Figure 1. However, for users, it is difficult to choose the right model for their NLP tasks, especially when there are no ground truth labels as a reference. Consequently, users tend to default to the most powerful LLM, resulting in unnecessarily high costs. To improve the accuracy-cost tradeoff, the variation in costs and quality among LLMs warrants a strategic approach to selecting the most appropriate model for a given task. For that, we introduce SpareLLM (**S**electing **P**assable **A**nd **R**esource-**E**fficient **LLM**s), a framework aimed at improving the efficiency of inference while providing equivalence guarantees.

Our goal is to minimize the cost of using LLMs for large-scale inference while preserving output quality. For certain NLP tasks, smaller models can often deliver results that are nearly identical to those of larger models. Specifically, we focus on selecting the least expensive model that produces outputs equivalent to the most powerful LLM, with a probability exceeding a user-defined threshold and confidence level. We refer to this most powerful LLM as the *reference LLM*. SpareLLM takes as input a task-specific question, a set of input instances, a user-defined equivalence constraint, and a confidence level. Given a collection of LLMs with varying performance and costs, SpareLLM generates outputs that match the reference LLM, ensuring equivalent outputs with a probability that meets the specified equivalence and confidence level. Later, we demonstrate that this equivalence guarantee also establishes a lower bound on the accuracy of the outputs of SpareLLM.

SpareLLM comprises two main phases: profiling and application. The profiling phase aims to gather information on the equivalence of each LLM by comparing its outputs to those of the reference model. Once sufficient information is collected, SpareLLM transitions to the application phase, where LLMs that meet (or closely approach) the equivalence criteria are used to process the remaining instances in a cost-efficient manner. Profiling comes with a cost because it involves processing the same item with multiple LLMs, including the reference model. Therefore, SpareLLM employs a strategic approach to choose the right amount of profiling, balancing the costs of profiling against the anticipated savings in the application phase.

In the profiling phase, SpareLLM gathers data on each LLM to assess its equivalence for the given NLP task. The aim is to identify LLMs whose outputs consistently align with those of the reference LLM, adhering to the user-defined equivalence constraint at the specified confidence level. To achieve this, SpareLLM executes all available LLMs on the same input instance and compares their outputs with that of the reference LLM. SpareLLM then iteratively accumulates more data by processing more items. This process is modeled as a series of Bernoulli trials, allowing SpareLLM to calculate binomial proportion confidence intervals at the given confidence level for the equivalence of each LLM. If the lower confidence interval bound of an LLM meets or exceeds the equivalence threshold, it is considered to have satisfied the equivalence constraint. This LLM can then be used in place of the reference LLM for subsequent LLM inferences during the application stage. Conversely, if the upper confidence interval bound falls below the equivalence threshold, it is deemed unsuitable for use. Profiling stops when the cheapest LLM that satisfies the equivalence constraint is found.

Profiling can be costly as it processes the same item with multiple LLMs. To address this, SpareLLM employs an early termination criterion, halting the profiling phase early if additional profiling is anticipated to be wasteful. SpareLLM assesses the benefit of additional profiling by weighing the extra costs of evaluating more items against the increased savings during the application phase. Estimating these expected costs is non-trivial, given that the true equivalence of each LLM remains unknown. SpareLLM only observes samples of input instances, knowing how many of the processed items are identical to those of the reference LLM. Hence, SpareLLM models the true equivalence as a binomial proportion under the binomial distribution. Based on this probabilistic approach, SpareLLM calculates the expected cost of profiling additional items. If the anticipated costs of further profiling are projected to exceed its benefits, SpareLLM halts the profiling process and moves on to the application phase.

A straightforward approach to the application phase involves employing the most affordable LLM that still meets the equivalence guarantees. However, further cost reductions can be achieved by leveraging a combination of multiple LLMs, even those whose individual equivalence does not meet the equivalence constraint. By strategically combining more affordable, less accurate LLMs with more expensive, more accurate ones, it is possible to further maximize cost savings. At the start of the application phase, SpareLLM distributes the remaining inference workload over available LLMs to minimize the overall cost, while adhering to the user-defined equivalence constraint. SpareLLM formulates this minimization problem as a mixed integer linear program (MILP), incorporating two key constraints: equivalence and confidence level. The equivalence constraint ensures that the collective equivalence across all inputs meets the given criteria. Similarly, the confidence level constraint requires that the confidence level aggregated from all deployed LLMs reaches the specified minimum. The solution to this MILP problem determines the ratios of the remaining input instances each LLM should process during the application phase.

In our experimental evaluation, we measure the cost savings of SpareLLM on NLP tasks using five real-world datasets: MMLU [14], IMDB [24], SMS-Spam [4], AgNews [43], and HellaSwag [41]. We employ multiple OpenAI LLMs, including gpt-4-turbo-2024-04-09, gpt-3.5-turbo-instruct, gpt-3.5-turbo-1106, davinci-002, and babbage-002. Compared to GPT-4-Turbo, with an equivalence constraint of $\geq$ 90%, SpareLLM achieves cost savings of 1.2×, 8.6×, 4.5×, 7.2×, and 1.4× for MMLU, IMDB, SMS-Spam, AgNews, and HellaSwag, respectively. Compared to two recent baselines, LLMCascade [40] and FrugalGPT [8], SpareLLM offers a superior cost-accuracy tradeoff across all benchmarks, dominating 91.1% of the points on the Pareto curve. We also experiment with Llama models (3B, 8B, 70B, 405B), where SpareLLM dominates 83.8% of the Pareto curve. In summary, the original contributions of this paper are as follows:

- We introduce SpareLLM, a novel framework that minimizes the cost of LLM inference tasks while providing equivalence guarantees. This enables users to trade result quality for reduced costs.
- We present a profiling scheme to evaluate the equivalence of LLMs in comparison to a reference LLM, along with an early termination criterion that effectively balances profiling overheads against future savings.
- We detail our method for strategically leveraging multiple LLMs with varying performance and costs, maximizing cost savings while meeting an equivalence constraint.
- We empirically evaluate SpareLLM on real-world datasets using OpenAI and Llama models, demonstrating significant cost savings and its ability to adapt to various equivalence targets. We compare SpareLLM to two recent baselines, LLMCascade and FrugalGPT, highlighting its superior cost-accuracy tradeoff.

The remainder of this paper is organized as follows. Section 2 introduces our formal problem model and Section 3 gives an overview of the SpareLLM framework. Section 4 describes the profiling phase of SpareLLM in more detail. Section 5 describes the early termination criterion for profiling. Section 6 describes the application phase of SpareLLM. We report experimental results in Section 7. We discuss related work in Section 8 before we conclude in Section 9.

2 Formal Model

SpareLLM targets the following scenario.

Target Scenario (LLMs with Varying Performance and Costs)**.** The performance of LLMs varies according to their sizes [15, 20]. Service providers often offer a range of LLMs with varying sizes, leading to differences in performance and cost.

The costs of higher-performing LLMs increase substantially with their performance levels. For instance, OpenAI charges \$0.0015 per input token for GPT-3.5 (turbo-0613), compared to \$0.03 for GPT-4 (0613), resulting in a cost difference of 20×. Consequently, users find it challenging to decide which LLM to use for a specific task, especially in common scenarios where ground truth labels are not available as a benchmark. Using the most expensive LLM is the safest choice in terms of output quality but incurs significant costs. On the other hand, opting for a cheaper and smaller LLM might render the outputs unusable.

Using large language models is expensive. However, there are tasks where significantly smaller models generate almost equivalent output to larger ones. Our goal is to select the cheapest model that generates equivalent output to the reference model (i.e., the model whose output is assumed to be the most reliable, typically the most expensive model at the same time) with a probability above a user-defined threshold with a user-specified confidence level.

Definition 1 ($\langle\delta, \gamma\rangle$-Equivalence)**.** Given a reference LLM $\overline{m}$, an LLM framework is $\langle\delta, \gamma\rangle$-equivalent if it generates outputs identical to those of the reference LLM with a probability of $1 - \delta$ at a confidence level of γ.

SpareLLM solves the following LLM inference problem.

Definition 2 (Task-Specific LLM Inference with $\langle\delta, \gamma\rangle$-Equivalence Guarantees)**.** Given a question q, inputs I, a user-defined equivalence constraint δ, and a confidence level γ, our goal is to generate outputs O by leveraging LLMs M with varying performance and costs. These outputs are equivalent to those of the reference LLM $\overline{m}$ in $100 \cdot (1 - \delta)\%$ of cases at a confidence level of γ.

As stated above, SpareLLM produces outputs with equivalence guarantees. We distinguish between the equivalence metric and the accuracy metric. Equivalence is measured relative to the outputs of the reference LLM, whereas accuracy is measured against the (hidden) ground-truth labels. By leveraging our equivalence guarantees, we derive a lower bound on the accuracy of the outputs of SpareLLM in relation to the accuracy of the reference LLM $\overline{m}$. The proof follows directly from the fact that the outputs O of SpareLLM are equivalent to those of the reference LLM $\overline{m}$ in $100 \cdot (1 - \delta)\%$ of cases.

Theorem 1 (Accuracy Bounds of SpareLLM)**.** The outputs O of SpareLLM with $\langle\delta, \gamma\rangle$-equivalence guarantees differ in accuracy from the reference LLM $\overline{m}$ by at most $100 \cdot \delta\%$ at a confidence level of γ. In other words, if the accuracy of the reference LLM $\overline{m}$ is $\overline{\kappa}$, the accuracy κ of SpareLLM is bounded by $|\overline{\kappa} - \kappa| \leq 100 \cdot \delta$ with confidence γ.

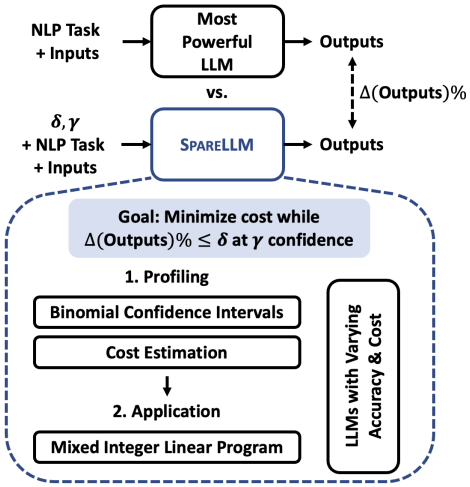


Fig. 2. Overview of SpareLLM.

// Generate outputs O for a question q from inputs I using LLMs M, ensuring that at least $100 \cdot (1-\delta)\%$ of the outputs are consistent with those from the reference LLM $\overline{m}$ with a confidence level of γ.

1: **function** SpareLLM($M, \overline{m}, q, I, \delta, \gamma$)
   // Profile equivalence of LLMs.
2: $\langle M, O\rangle \leftarrow$ Profile($M, \overline{m}, q, I, \delta, \gamma$)
   // Calculate the ratio of items processed during profiling.
3: $r \leftarrow |O|/|I|$
   // Remove items processed during profiling.
4: $I \leftarrow \{i \in I | \nexists o : \langle i, o\rangle \in O\}$
   // Process remaining items based on LLM profiles.
5: $O \leftarrow O \cup$ Apply($M, q, I, \delta, \gamma, r$)
6: **return** O

Algorithm 1. The SpareLLM framework.

3 *SpareLLM* Framework

Figure 2 presents an overview of the SpareLLM framework. SpareLLM is designed to minimize the cost of LLM inference on a large number of inputs for a given task, while satisfying user-defined equivalence constraints. At its core, the framework features a task-specific selection of LLMs with equivalence guarantees. We provide equivalence guarantees of outputs against the reference model, which is typically the most powerful and expensive LLM.

Algorithm 1 describes SpareLLM at a high level of abstraction. It takes a question q, a set of inputs I, a reference LLM $\overline{m}$, a set of LLMs with varying costs M, a target equivalence $1-\delta$, and a confidence level γ. SpareLLM produces outputs O for the question q on inputs I, ensuring that the outputs differ from those of the reference model in no more than $100 \cdot \delta\%$ of cases with confidence γ. SpareLLM consists of two phases: profiling and application. The main purpose of profiling is to estimate the probability of output equivalence for each LLM relative to the reference LLM. Based on this information, we process the remaining items (after profiling) to maximize cost savings while satisfying the user-defined equivalence constraint.

We present three variants of SpareLLM, each adding a new feature to the preceding variant: 1) ProfileAll, 2) ProfileSmart, and 3) ModelMix. All versions of SpareLLM share the same

Table 1. LLM m and its attributes.

Attribute	Semantics
$m.n$	Number of processed items using m
$m.e$	Number of processed items whose outputs are equivalent to those of the reference model $\overline{m}$
$m.c$	Average cost of processing a single item using m, which is updated as more items are processed
$m.s$	Status of m: `Unknown`, `Valid`, `Invalid`

// Profile LLMs M by evaluating whether their outputs for question q from inputs I are equivalent to those of the reference LLM $\overline{m}$ until sufficient information is obtained. The stopping criterion is based on the equivalence constraint δ and confidence level γ.

function PROFILE($M, \overline{m}, q, I, \delta, \gamma$)
  // Initialize status of each LLM to unknown.
  $\forall m \in M : m.s \leftarrow$ `Unknown`
  // Update status of the reference LLM to valid.
  $\overline{m}.s \leftarrow$ `Valid`
  // Start profiling and store outputs of the reference LLM.
  $O \leftarrow \emptyset$
  for all $i \in I$ **do**
    // Process item with the reference LLM $\overline{m}$.
    $\overline{o} \leftarrow \overline{m}(q, i)$
    $O \leftarrow O \cup \{\langle i, \overline{o}\rangle\}$
    // Process item with cheaper LLMs and check equivalence.
    for all $m \in \{m' \in M | m'.s =$ `Unknown`$\}$ **do**
      $m.n \leftarrow m.n + 1$
      $o \leftarrow m(q, i)$
      if $o = \overline{o}$ **:**
        $m.e \leftarrow m.e + 1$
    // Update LLM status based on equivalence of their outputs.
    $M \leftarrow$ EVALMODELS(M, δ, γ)
    // Number of remaining items to process.
    $n \leftarrow |\{i \in I | \nexists o : \langle i, o\rangle \in O\}|$
    // Stop profiling if condition is met.
    if TERMINATEPROFILE($M, \overline{m}, \delta, \gamma, n$) **:**
      break
  return $\langle M, O\rangle$

Algorithm 2. Profile LLMs to find models satisfying the equivalence constraint.

high-level structure, as shown in Algorithm 1. The first version, SPARELLM-PROFILEALL, introduces the profiling phase, where we collect information on the equivalence of each available LLM. After profiling, it selects the most cost-efficient LLM, which is believed to meet the equivalence constraint with adequate confidence, to process the remaining items. The second version, SPARELLM-PROFILESMART, builds upon the first by evaluating the potential savings from profiling and by terminating profiling early when further profiling is considered wasteful. The final version, SPARELLM-MODELMIX, enhances the application phase by selecting a combination of multiple models that together satisfy the user-defined equivalence constraints for processing the remaining items. We provide more details in the following sections.

// Given profiled LLMs M, terminate profiling if a valid LLM is as cheap as any LLM with unknown status.
1: **function** TerminateProfile[ProfileAll](M, _, _, _, _)
// Find valid LLM with the smallest cost.
2: $\underline{m} \leftarrow \arg\min_{\{m \in M | m.s = \texttt{Valid}\}} m.c$
// Terminate if a valid LLM is as cheap as any unknown LLM.
3: **if** $\underline{m}.c \leq \min_{\{m \in M | m.s = \texttt{Unknown}\}} m.c$ **:**
4: **return** `True`
5: **return** `False`

Algorithm 3. Terminate profiling if found cheapest LLM that satisfies the equivalence constraint.

// Evaluate the status of each LLM $m \in M$ by calculating binomial confidence intervals with confidence γ, based on the number of processed items $m.n$ and the number of items with equivalent outputs $m.e$. An LLM is invalid if its equivalence upper bound falls below the equivalence threshold $1 - \delta$, and valid if its lower bound meets or exceeds the threshold.
1: **function** EvalModels(M, δ, γ)
2: **for all** $m \in \{m' \in M | m'.s = \texttt{Unknown}\}$ **do**
// Compute binomial confidence intervals.
3: $\langle l, u \rangle \leftarrow$ BinomCI($m.n, m.e, \gamma$)
// LLM is invalid if upper bound equivalence is too low.
4: **if** $u < 1 - \delta$ **:**
5: $m.s \leftarrow$ `Invalid`
// LLM is valid if lower bound equivalence is sufficiently high.
6: **if** $l \geq 1 - \delta$ **:**
7: $m.s \leftarrow$ `Valid`
8: **return** M

Algorithm 4. Evaluate models as valid or invalid based on binomial confidence intervals.

4 *ProfileAll*: Profiling Models Exhaustively

The key feature of SpareLLM-ProfileAll is the profiling phase. SpareLLM-ProfileAll profiles all LLMs to determine whether each model meets or falls short of the user-defined equivalence threshold at the specified confidence level. For that, we introduce three possible status values for LLMs based on their current profile. A model is considered `Valid` if we have sufficient information to be confident that it meets the equivalence constraint. A model is deemed `Invalid` if we are confident that it fails to meet the equivalence constraint. The status of an LLM is `Unknown` if there is not enough information to either accept or reject the LLM as satisfying the equivalence constraint. The profiling process ends when there is a `Valid` LLM as cost-efficient as any LLM with `Unknown` status. That is, we have identified the most affordable LLM that meets the equivalence constraint.

Algorithm 2 illustrates the profiling phase in more detail. First, we initialize the status of each LLM (except the reference LLM) to `Unknown`. Next, we evaluate input items using the reference model as well as the other cheaper LLMs with `Unknown` status. We compare their outputs and keep track of the number of items processed and the number of items whose outputs are equivalent to those of the reference model. Then, we update the status of each LLM to either `Valid` or `Invalid` if we have gathered sufficient information using the `EvalModels` function, as presented in Algorithm 4. We provide more details in the following paragraph. Lastly, we stop profiling, when the termination criterion described in Algorithm 3 is met. Profiling is terminated based on the current profiles of LLMs, specifically when there is a `Valid` LLM whose cost is lower than or equal to that of any LLM with `Unknown` status. This termination condition guarantees that we have identified the most

```
// Given profiled LLMs M, question q, and remaining inputs I, process remaining items using the cheapest
LLM that satisfies the equivalence constraint.
1: function Apply[ProfileAll, ProfileSmart](M, q, I, _, _, _)
    // Find cheapest LLM satisfying equivalence constraint.
2:  m ← arg min_{m∈M|m.s=Valid} m.c
    // Process remaining items.
3:  O ← ∅
4:  for all i ∈ I do
5:      o ← m(q, i)
6:      O ← O ∪ {⟨i, o⟩}
7:  return O
```

Algorithm 5. Single-model application.

cost-efficient LLM satisfying the user-defined equivalence constraint. Table 1 summarizes the LLM profile properties, used in Algorithm 2 and the following algorithms.

Algorithm 4 describes the process of evaluating the status of each LLM m by calculating the binomial confidence interval for its equivalence. Here, we assume running an LLM on instances is a Bernoulli process, where the output of an instance could either match or differ from that of the reference model. The BinomCI function in Algorithm 4 employs the Clopper-Pearson exact method [10] to compute the binomial confidence intervals for LLM m. We denote the lower and upper bounds of the confidence interval by l and u, respectively. If the upper bound of the model equivalence falls below the equivalence threshold (i.e., $u < 1-\delta$), the model m is considered `Invalid` for processing the remaining items. In contrast, if the lower bound of the model equivalence is higher than or equal to the equivalence threshold (i.e., $l \geq 1-\delta$), then the status of m is updated to `Valid`. If neither of these conditions is satisfied, we continue profiling m in the subsequent iteration. Figure 9 in the experimental section illustrates the development of confidence intervals on the equivalence of each LLM during the profiling phase.

Next, Algorithm 5 outlines the application phase of SpareLLM-ProfileAll. During this phase, we select the most cost-effective LLM among those identified as `Valid` (which may include the reference LLM). We then process each of the remaining items using the selected LLM. This procedure ensures that we leverage a cost-efficient solution while adhering to the equivalence guarantees established during the profiling phase. As described in the last line of Algorithm 1, the outputs generated during this phase are combined with the outputs from the profiling phase and are presented to the user. We denote different versions of algorithms for SpareLLM variants using square brackets following the function name, as shown in Algorithm 5.

We briefly discuss extensions to NLP tasks where a comparison of outputs for exact match (as in line 11 of Algorithm 2) is misleading. For example, in text summarization, the likelihood of different LLMs producing exactly identical outputs is quite small. One potential solution involves using a more sophisticated metric, such as a distance function comparing two texts, to assess the equivalence of the outputs. However, implementing a non-binary scoring function would require redesigning the framework, as it currently employs a Bernoulli process model. As an alternative, we can map a numerical metric to a binary result using a threshold. Another solution might be to utilize an LLM to compare the two outputs for semantic matching, making it applicable to any open-ended generation task.

5 *ProfileSmart*: Restricting Profiling Overheads

SpareLLM-ProfileSmart introduces a feature that terminates profiling early if further evaluation is expected to be wasteful. Profiling involves running both the reference LLM, which is typically the

// Given profiled LLMs M, reference LLM $\overline{m}$, equivalence constraint δ, confidence level γ, and the remaining n items, terminate profiling if further profiling is expected to be wasteful (or a valid LLM is as cheap as any LLM with unknown status).

function TerminateProfile[ProfileSmart]($M, \overline{m}, \delta, \gamma, n$)
  // Terminate if a valid LLM is as cheap as any unknown LLM.
  if TerminateProfile[ProfileAll](M) **:**
    return `True`
  // Find current cheapest LLM satisfying equivalence constraint.
  $\underline{m} \leftarrow \arg\min_{\{m \in M | m.s = \texttt{Valid}\}} m.c$
  // Compute the expected cost if profiling terminates at this point.
  $\underline{c} \leftarrow n \cdot \underline{m}.c$
  // Store expected costs for varying numbers of profiled items.
  $k = 1$
  while $k \leq n$ **:**
    // Compute the expected cost if profiling exactly k more items.
    $c_k \leftarrow \text{Cost}(k, M, \overline{m}, \underline{m}, \delta, \gamma, n)$
    $k \leftarrow 2 \cdot k$
  // Terminate if further profiling is expected to be wasteful.
  if $\underline{c} \leq \min_k c_k$ **:**
    return `True`
  return `False`

Algorithm 6. Terminate profiling if further profiling is expected to be wasteful.

most expensive, and the other LLMs on the same input instance. Our hope is that the cost of using the reference LLM is offset by the savings from employing a less expensive LLM for processing the remaining items after profiling. However, profiling costs can outweigh the cost savings when a large number of items are needed to validate an LLM with `Unknown` status as either `Valid` or `Invalid`. Therefore, we estimate the expected cost savings from profiling additional items and terminate early if the projected savings are not positive.

For example, suppose a user applies an equivalence constraint of $1 - \delta = 0.9$, and the true equivalence of one of the cost-efficient LLMs is also 0.9 (which is unknown during profiling). In this case, SpareLLM may be unable to validate this LLM as either `Valid` or `Invalid` for a large number of profiled items, as neither the lower nor the upper bound of the confidence interval meets the equivalence threshold of $1 - \delta = 0.9$. By terminating profiling early, SpareLLM-ProfileSmart avoids excessive profiling when further effort is unlikely to benefit the application phase.

Algorithm 6 illustrates the newly added termination criterion based on expected cost savings. First, we calculate the expected cost of using the most cost-efficient LLM among the current set of `Valid` LLMs to process the remaining items. For that, we multiply the number of remaining items by the unit cost (refer to $m.c$ in Table 1) of the cheapest `Valid` LLM. This serves as a baseline, assuming we cease profiling at this juncture and proceed to the application phase. Next, we assess the impact of profiling more items on the overall cost. Specifically, we calculate the expected costs for different values of k, where k represents the number of additionally profiled items. We do not know a priori which number of profiled items will yield the optimal balance between profiling overheads and the cost savings in the application phase. Hence, we start with $k = 1$ and incrementally double its value up to the number of remaining items. This exponential scheme is reasonable, given that profiling too many items would be wasteful, and the search should concentrate on the range with smaller values. Finally, we compare the lowest among the costs of profiling additional items against the cost of halting profiling at this point. We terminate profiling if setting $k = 0$ minimizes expected costs.

The Cost function in Algorithm 6 calculates the expected cost of profiling LLMs for exactly k more items, followed by the application phase. That is, we continue profiling for k additional items and then process remaining items based on the newly collected information. To compute the expected cost, we first need to determine for each LLM m with Unknown status (i.e., $m.s$ = Unknown) whether m will eventually be considered Valid after profiling for k more items. We express the probability that LLM m will be evaluated as Valid as the following:

$$\Pr(m.s = \texttt{Valid} | k, \delta, \gamma)$$

Next, we assign a sort order to LLMs based on their unit costs (since the same prompt is used for all LLMs, this effectively amounts to ordering the LLMs based on their per-token costs). Among the LLMs with Unknown status, we label the LLM with the smallest unit cost as m_1, the one with the second smallest unit cost as m_2, and so on. Based on this ordering, if m_1 is evaluated to be Valid, it will be used during the application phase due to its smallest unit cost. If m_1 is not Valid, then the LLM m_2 with the next smallest unit cost will be considered. Namely, this ordering represents the sequence in which the LLMs will be considered during the application stage. Lastly, assuming independence between LLMs, we can formulate the expected cost as follows. For simplicity, we denote the probability that LLM m_i is evaluated as Valid by $p_i = \Pr(m_i.s = \texttt{Valid} | k, \delta, \gamma)$.

$$\begin{aligned} \textsc{Cost}(k, M, \overline{m}, \underline{m}, \delta, \gamma, n) = k(\overline{m}.c + \sum_i m_i.c) \\ + (n-k) \sum_i \Big(\prod_{j<i} (1-p_j)\Big) \cdot p_i \cdot m_i.c \\ + (n-k)\Big(\prod_i (1-p_i)\Big) \cdot \underline{m}.c \end{aligned} \tag{1}$$

The first term, $k(\overline{m}.c + \sum_i m_i.c)$, represents the cost of profiling k additional items using the reference LLM $\overline{m}$ alongside the LLMs with Unknown status $\{m_i\}$. The second term accounts for the cost of processing the remaining $(n-k)$ items using the cheapest LLM among those newly verified as Valid. The third term addresses the scenario where none of the LLMs are evaluated to be Valid and thus rendering the profiling wasteful. By summing these terms together, we derive the expected cost of profiling exactly k more items and processing the remaining items in the application phase.

To calculate the expected cost formula, we need to determine the probability of an LLM being Valid after profiling k more items. An LLM is evaluated as Valid if the lower bound of the binomial confidence interval on its equivalence is greater than or equal to the equivalence threshold $1-\delta$ with confidence γ. Thus, we obtain:

$$\Pr(m.s = \texttt{Valid} | k, \delta, \gamma) = \Pr(l_m \geq 1 - \delta)$$

where l_m is the lower bound of the binomial confidence interval (with confidence γ) for LLM m after profiling k additional items. We now want to compute the binomial confidence interval after profiling k more items. Recall that profiling an LLM m reveals that after $m.n$ items are processed, $m.e$ items produce outputs equivalent to the reference model. Based on this current information, the number of processed items is calculated as the sum of k and the number of currently profiled items $m.n$. For LLM m, among the k newly profiled items, we denote the number of items with the same outputs as the reference model by e_m. Similar to the number of processed items, the number of conforming items is given as the sum of e_m and $m.e$. The confidence level is given by γ. Thus, the lower bound l_m of the binomial confidence interval is derived from the function: $\textsc{BinomCI}((k + m.n), (e_m + m.e), \gamma)$. As a result, we can solve the inequality $l_m \geq 1 - \delta$ for e_m to find the smallest value of e_m (which we denote by e_m^*) that satisfies the following condition:

$$\textsc{BinomCI}((k + m.n), (e_m + m.e), \gamma).\textsc{Lower} \geq 1 - \delta$$

By treating the number of conforming items e_m as a random variable $\mathbf{e}$, we rewrite the probability of an LLM being `Valid` after profiling k additional items as:

$$\Pr(m.s = \texttt{Valid} | k, \delta, \gamma) = \Pr(l_m \geq 1 - \delta) = \Pr(\mathbf{e} \geq e_m^*)$$

Random variables are represented by symbols in bold font. If given the true equivalence of LLM m when evaluated on all items as a, we can compute this probability using the binomial distribution. The random variable $\mathbf{e}$ follows the binomial distribution with parameters k and a as: $\mathbf{e} \sim \textsc{Binom}(k, a)$. Therefore, by denoting the cumulative distribution function (CDF) of this binomial distribution as $B(x) = \Pr(\mathbf{e} \leq x)$, we get the probability as:

$$\Pr(\mathbf{e} \geq e_m^* | a) = 1 - B(e_m^* - 1)$$

However, the challenge is that we do not know the true equivalence a of LLM m. We instead rely on the observations from profiling, $m.n$ and $m.e$, to estimate the true equivalence as $\hat{a} = m.e/m.n$. Then, under the same assumptions as those used in the Wald interval (based on the central limit theorem) [5], we approximate the estimation error using a Gaussian distribution as follows:

$$\frac{\hat{a} - a}{\sqrt{\hat{a}(1 - \hat{a})/m.n}} \sim \textsc{Norm}(0, 1)$$

That is, the random variable $\mathbf{a}$, which models the true equivalence a, follows a Gaussian distribution as the following:

$$\mathbf{a} \sim \textsc{Norm}(\mu = \hat{a}, \sigma^2 = \frac{\hat{a}(1 - \hat{a})}{m.n})$$

The intuition behind this formalization is that as we collect more information (i.e., profile more items), the variance of the error decreases due to the term $\frac{1}{m.n}$. In other words, as the sample size increases, it becomes more likely that the sample mean $\hat{a}$ accurately represents the true equivalence a.

Finally, we compute the probability of LLM m being `Valid` after processing k additional items by integrating over the possible range of $a \in [0, 1]$. By denoting the probability density function (PDF) of this normal distribution as $f(x) = \Pr(\mathbf{a} = x)$, we obtain:

$$\begin{aligned}\Pr(\mathbf{e} \geq e_m^*) &= \int_0^1 \Pr(\mathbf{e} \geq e_m^* | a) \Pr(a)\, da \\ &= \int_0^1 (1 - B(e_m^* - 1)) f(a)\, da\end{aligned}$$

where $B(x)$ is the CDF of $\textsc{Binom}(k, a)$ and $f(x)$ is the PDF of $\textsc{Norm}(\hat{a}, \frac{\hat{a}(1-\hat{a})}{m.n})$. We can now compute p_i in Equation 1 and use the function to calculate expected costs. Figure 10 in the experimental section illustrates the expected costs of profiling additional items, including the profiling overheads and the savings anticipated in the application phase.

We briefly discuss the impact of the independence assumption from Equation 1 on the effectiveness of SpareLLM. In practice, LLMs can be positively correlated, meaning for instance that the cheapest model producing the reference answer for an intrinsically "easy" task implies that other models produce the reference answer as well. If models have complementary strengths, negative correlation (meaning that one model producing the reference answer makes other models less likely to do so) is also possible. In such cases, SpareLLM misses opportunities to gain information due to the independence assumption, possibly increasing profiling phase unnecessarily. However, an unjustified independence assumption does not impact the equivalence guarantees of SpareLLM.

// Given profiled LLMs M, question q, remaining inputs I, and the ratio of items processed during profiling r, process remaining items using multiple LLMs to minimize cost while satisfying the equivalence constraint δ with a confidence level γ.

function Apply[ModelMix]($M, q, I, \delta, \gamma, r$)
  // Find ratio per LLM.
  $\{r_m\}_{m \in M} \leftarrow$ ComputeRatios(M, δ, γ, r)
  // Split remaining items based on ratios.
  $\{I_m\}_{m \in M} \leftarrow$ PartitionByRatios($I, \{r_m\}$)
  // Process remaining items.
  $O \leftarrow \emptyset$
  for all $m \in M$ **do**
    for all $i \in I_m$ **do**
      $o \leftarrow m(q, i)$
      $O \leftarrow O \cup \{\langle i, o \rangle\}$
  return O

Algorithm 7. Multiple-models application.

The application phase of SpareLLM-ProfileSmart is identical to that of SpareLLM-ProfileAll. As described in Algorithm 5, we select the `Valid` LLM with the lowest unit cost and apply it to the remaining items. The distinction between the two variants of SpareLLM lies in the profiling phase, where SpareLLM-ProfileSmart carefully evaluates the tradeoff between profiling overheads and application savings.

6 *ModelMix*: Selecting Model Combinations

SpareLLM-ModelMix improves the application phase by leveraging all LLMs to maximize cost savings. Previously, in Algorithm 5, we employ a single LLM to process the remaining items in the application phase. This earlier method misses a potential opportunity for additional cost savings. Consider an LLM that is more cost-efficient but has an equivalence lower bound just below the equivalence threshold. Such slight shortfall in equivalence disqualifies the LLM as a valid option for application in the previous method. However, recognizing that its equivalence is close to the user-defined threshold, SpareLLM-ModelMix proposes a novel approach. The main idea involves combining this LLM with an LLM with a higher equivalence. By partitioning the processing of remaining items between the more economical LLM and a more expensive, higher-equivalence LLM, we can realize further cost reductions.

Algorithm 7 describes the novel approach for the application phase. At its heart is the `ComputeRatio` function, which determines the percentage of remaining items that each LLM should process. This calculation is based on solving a mixed integer linear program, with further details provided in the subsequent paragraphs. We then partition the remaining items according to the determined ratios and process them using the respective LLMs. Due to this careful partitioning, we ensure that the generated outputs, in aggregate, meet the equivalence constraint at the designated confidence level.

We provide a detailed explanation of the `ComputeRatio` function. Our objective is to minimize the cost of processing the remaining items while adhering to the equivalence constraint. To achieve this, we focus on minimizing the cost function below, where i indexes all available LLMs:

$$\min \sum_i c_i x_i$$

where c_i is the unit cost of an LLM m_i and $x_i \in [0, 1]$ is the ratio of items processed by model m_i. The ratios x_i should sum up to one: $\sum_i x_i = 1$.

There are two key constraints associated with this minimization problem: 1) equivalence constraint and 2) confidence level constraint. We need to guarantee that, after processing all items, we satisfy the equivalence threshold $1-\delta$ with confidence γ. To specify these constraints, we first introduce a separate confidence level γ_i for each LLM m_i. The main idea is that the combined confidence of these individual confidence levels γ_i should be greater than or equal to the confidence constraint γ. As a result, assuming independence between confidence levels, we get the confidence level constraint as follows:

$$\prod_i \gamma_i \geq \gamma \tag{2}$$

Here, without loss of generality, we say that LLMs (except for the reference model) exhibit zero equivalence lower bound for a 100% confidence level.

Then, we specify the equivalence constraint based on the equivalence lower bound of each LLM m_i. First, we compute the lower bound l_i on the equivalence of LLM m_i, using the binomial confidence intervals with confidence γ_i. That is, for each LLM m_i, given the number of processed items, $m_i.n$, and the number of items whose outputs conform with the reference LLM, $m_i.e$, we calculate the lower bound from the function $\textsc{BinomCI}(m_i.n, m_i.e, \gamma_i)$. Now, we need to guarantee that the final equivalence satisfies the given equivalence constraint. In other words, we want the accumulated equivalence over all items to be at least the same as the equivalence threshold. We temporarily denote this equivalence threshold by α instead of the previously used $1-\delta$ (since its value slightly differs from $1-\delta$, a distinction we will explain shortly). Given the equivalence lower bound l_i per LLM m_i, we specify the equivalence constraint as:

$$\sum_i l_i x_i \geq \alpha \tag{3}$$

When computing the final equivalence over all items, we should also take into account the items that we processed during the profiling phase (which we processed using the reference LLM). In that, we use a slightly refined version of the equivalence threshold (instead of $\alpha = 1-\delta$) in this optimization problem. Note that the equivalence of the reference LLM is one, as we define equivalence in terms of consistency with the outputs of the reference model. Denoting by r the ratio of items processed in the profiling phase to the total number of items, we solve the following equation to get the refined equivalence threshold α:

$$1 \cdot r + \alpha(1-r) = 1-\delta \Rightarrow \alpha = 1 - \frac{\delta}{1-r}$$

We solve this minimization problem for x_i to find the optimal ratios for distributing the remaining items to LLMs. However, solving this problem directly is difficult due to the non-elementary natural of computing quantiles (e.g., Gauss error function if using a normal approximation). That is, it is hard to express the equivalence lower bounds l_i as an elementary function of other variables. Instead, we define a fixed set of potential confidence levels $\{\gamma_j\}$ where each level is indexed by j (instead of i). Then, we introduce a binary variable $y_{ij} \in \{0,1\}$ that indicates whether the LLM m_i operates under the confidence level γ_j. We assign at most one confidence level per LLM, thereby introducing the following constraint: $\forall i : \sum_j y_{ij} \leq 1$. In our implementation, we utilize a range of confidence levels starting from the user-defined confidence level γ up to one, with an increment of 0.01. For instance, if a user specifies a confidence level of 0.95, we consider potential confidence levels $\gamma_j \in \{0.95, 0.96, 0.97, 0.98, 0.99, 1\}$. Now, we reformulate the two constraints in Equations 2 and 3.

First, we rewrite the equivalence constraint in Equation 3. Since we specify predefined values $\{\gamma_j\}$ for confidence levels, we can compute the equivalence lower bound l_{ij} for each combination of

LLM m_i and confidence level γ_j. Now, we replace the term l_i in Equation 3 using the new variables: $l_i = \sum_j l_{ij} y_{ij}$. Note that, due to the constraint on y_{ij}, only one of y_{ij} can have a value of one for a fixed i. As a result, we reformulate Equation 3 as the following:

$$\sum_{ij} l_{ij} x_i y_{ij} \geq 1 - \frac{\delta}{1-r} \tag{4}$$

Second, we rewrite the confidence level constraint by replacing γ_i using the new variables γ_j and y_{ij}. The confidence level γ_i of LLM m_i can be expressed as: $\gamma_i = \prod_j (1-(1-\gamma_j)y_{ij})$. The term inside the product has a value of one when $y_{ij} = 0$ and otherwise γ_j when $y_{ij} = 1$. Thus, we reformulate Equation 2 as follows:

$$\prod_{ij} (1-(1-\gamma_j)y_{ij}) \geq \gamma \tag{5}$$

In order to formulate this minimization problem as a mixed integer linear program (MILP), we convert the product in Equation 5 into a summation. By applying logarithmic function to both sides, we obtain:

$$\sum_{ij} \ln(1-(1-\gamma_j)y_{ij}) \geq \ln \gamma$$

Since y_{ij} is binary, we have $\ln(1-(1-\gamma_j)y_{ij}) = 0$ if $y_{ij} = 0$ and $\ln(1-(1-\gamma_j)y_{ij}) = \ln \gamma_j$ if $y_{ij} = 1$. That is, we arrive at the following equality:

$$\ln(1-(1-\gamma_j)y_{ij}) = y_{ij} \ln \gamma_j$$

Therefore, we rewrite the confidence level constraint as:

$$\sum_{ij} y_{ij} \ln \gamma_j \geq \ln \gamma \tag{6}$$

In summary, based on the reformulated constraints in Equations 4 and 6, we solve the following MILP problem:

$$\min \sum_i c_i x_i$$

subject to

$$\sum_{ij} l_{ij} x_i y_{ij} \geq 1 - \frac{\delta}{1-r},$$

$$\sum_{ij} y_{ij} \ln \gamma_j \geq \ln \gamma,$$

$$\sum_i x_i = 1,$$

and

$$\forall i : \sum_j y_{ij} \leq 1$$

where $x_i \in [0,1]$ and $y_{ij} \in \{0,1\}$.

7 Experimental Evaluation

We describe the experimental setup in Section 7.1 and present the experimental results in Section 7.2.

Table 2. Overview of benchmarks.

Dataset	Task	#Instances	#Tokens
MMLU [14]	Multiple-Choice Question	14,042	128.0
IMDB [24]	Sentiment Classification	50,000	331.7
SMS-Spam [4]	Spam Detection	5,574	60.8
AgNews [43]	Topic Classification	127,600	95.9
HellaSwag [41]	Commonsense Inference	10,042	214.4

7.1 Experimental Setup

Benchmarks. We evaluate SpareLLM using five well-established real-world datasets, summarized in Table 2. The table provides details about the task, the number of instances, and the average number of prompt tokens per instance. First, the MMLU benchmark [14] is a multiple-choice question answering dataset that tests knowledge across various topics, such as mathematics, history, and law. We use the test set, consisting of 14,042 instances, for our evaluation. Following prior work [29, 33], we use the same prompt but apply zero-shot instead of few-shot learning. Second, the IMDB benchmark [24] is a widely recognized dataset for binary sentiment classification. It consists of 50,000 movie reviews from the Internet Movie Database (IMDB). For our evaluation, we utilize all instances from both the training and testing sets. Third, the SMS-Spam Collection benchmark [4] is a dataset collected for the purpose of spam detection in Short Message Service (SMS). It comprises a collection of messages labeled as spam or legitimate with a total of 5,574 instances. Fourth, the AgNews benchmark [43] is a dataset designed for news categorization tasks, containing 120,000 training samples and 7,600 test instances. Each entry is a news article categorized into one of four classes: World, Sports, Business, or Science/Technology. We again use all available instances for our evaluation. Fifth, the HellaSwag benchmark [41] is a natural language inference dataset designed to evaluate commonsense reasoning. We use the validation set, consisting of 10,042 instances, since the test set does not include ground-truth labels. Using the tokenizer for GPT-4-Turbo, on average, the numbers of tokens per instance are 128.0, 331.7, 60.8, 95.9, and 214.4 for MMLU, IMDB, SMS-Spam, AgNews, and HellaSwag, respectively.

Large Language Models. In our experiments, we employ a range of OpenAI [30] models as our available LLMs. Specifically, we use the following set of models: gpt-4-turbo-2024-04-09, gpt-3.5-turbo-instruct, gpt-3.5-turbo-1106, davinci-002, and babbage-002. The costs per 1,000 input tokens for these models are \$0.01, \$0.0015, \$0.001, \$0.002, and \$0.0004, respectively. We designate GPT-4-Turbo (gpt-4-turbo-2024-04-09) as the reference model for our experiments. In addition, we utilize Llama 3 models [12] hosted by the provider Together AI [3]. The following four model sizes are used: 3B, 8B, 70B, and 405B (with the 405B serving as the reference model). Their costs per 1,000 tokens are \$0.00006, \$0.00018, \$0.00088, and \$0.0035, respectively. We employ the same prompt template across all LLMs.

Baselines. We compare SpareLLM against two recently proposed LLM frameworks for cost-efficient inference: LLMCascade and FrugalGPT. LLMCascade [40] is a cascading pipeline that uses the consistency of the responses as a proxy for question difficulty. The core idea is to query an LLM multiple times on the same question to measure its answer consistency. The pipeline requires users to set a threshold for answer consistency, cascading to a stronger LLM if a weaker LLM fails to meet the threshold. For LLMCascade, we sort the five LLMs by their input token cost and create the cascading pipeline. As in [40], we vary the threshold from 0.4 (i.e., accept if 40% of responses are identical) to 1 (i.e., accept only if all responses are identical). FrugalGPT [8] is an LLM framework that also leverages model cascading for efficient inference. It enables users to specify a target cost

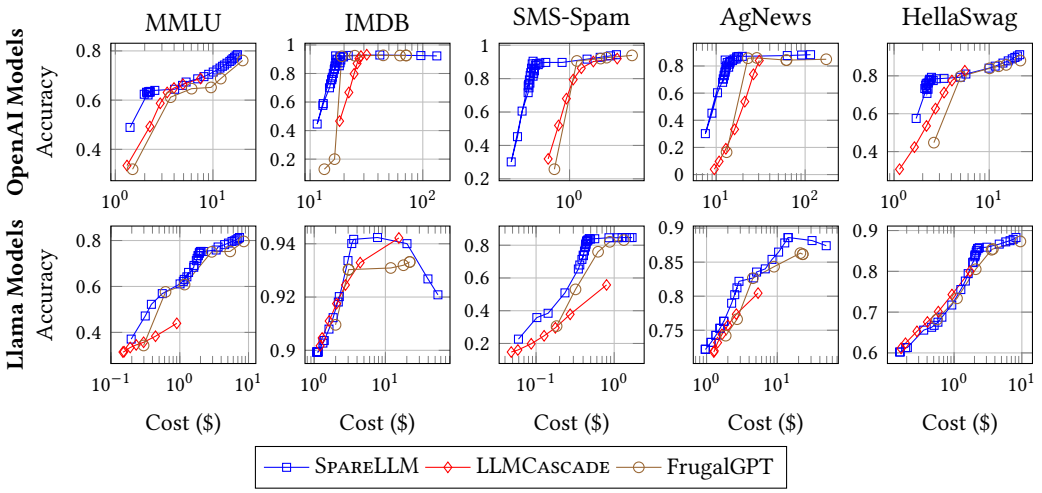


Fig. 3. Cost-accuracy tradeoff curves for SpareLLM compared to LLMCascade and FrugalGPT under various configurations using OpenAI models (top row) and Llama models (bottom row).

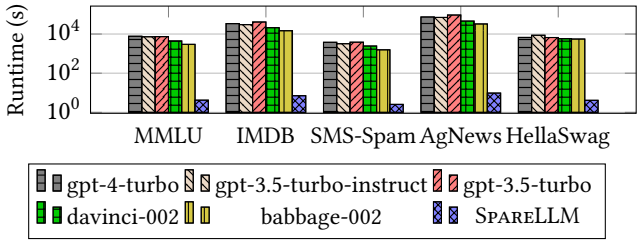


Fig. 4. Runtime overhead of SpareLLM compared to LLM inference on all benchmarks.

budget for processing a single input for the given NLP task. Using a training set with ground truth labels, it learns the score thresholds for each LLM in the cascading pipeline to satisfy the target cost budget. During inference, FrugalGPT returns the answer from the first LLM that satisfies the learned threshold. In our experiments, we provide FrugalGPT with 100 ground truth labels for training with no additional cost. In our ablation study, we evaluate all variants of SpareLLM, including SpareLLM-ProfileAll, SpareLLM-ProfileSmart, and SpareLLM-ModelMix. Given that SpareLLM-ModelMix represents the final variant of the framework, we subsequently refer to it as SpareLLM.

Evaluation Setup. To reduce evaluation costs, we evaluate all instances from the benchmarks on all LLMs and store their responses. In addition, we record the cost and response time for each instance. We then evaluate SpareLLM and the baselines using these stored responses. In line with its requirements, we collect multiple responses per input instance with a non-zero temperature to support LLMCascade. Consistent with prior work [40], we set the temperature to 0.4 and the number of responses to 20. Since the order of instances can influence the outcome, we run SpareLLM 10 times per setting with different random orderings and report the average values unless noted otherwise. Throughout the experiments, we use a confidence level of $\gamma = 0.95$. For the equivalence constraint δ, we test a wide range of settings from 0.02 to 0.9. For example, an equivalence constraint of $\delta = 0.02$ implies that SpareLLM produces outputs consistent with those of GPT-4-Turbo in 98% of cases with a 95% confidence level. Based on Theorem 1, this also indicates that the accuracy

Table 3. Minimum and mean equivalence (%) of SpareLLM compared to GPT-4 across 10 runs for varying equivalence thresholds $1-\delta$, along with the number of runs failing to meet the constraint.

	MMLU				IMDB				SMS-Spam				AgNews				HellaSwag		
$1-\delta$	Min	Avg	#F	$1-\delta$	Min	Avg	#F	$1-\delta$	Min	Avg	#F	$1-\delta$	Min	Avg	#F	$1-\delta$	Min	Avg	#F
0.96	95.6	97.5	1	0.98	96.6	98.7	1	0.98	98.3	99.1	-	0.98	98.8	99.1	-	0.98	98.3	99.1	-
0.92	91.4	94.9	1	0.96	93.6	97.3	2	0.96	96.4	97.8	-	0.96	96.5	97.7	-	0.96	96.8	98.0	-
0.88	87.8	92.3	1	0.94	93.9	95.2	1	0.94	93.6	96.1	1	0.94	95.1	95.9	-	0.94	95.1	96.8	-
0.84	84.0	88.8	1	0.92	93.0	94.2	-	0.92	91.6	94.4	2	0.92	93.1	93.2	-	0.92	93.4	95.7	-
0.8	81.1	86.1	-	0.9	92.7	93.6	-	0.9	90.7	91.4	-	0.9	92.9	93.0	-	0.9	91.9	94.2	-
0.76	77.0	82.4	-	0.88	92.5	93.2	-	0.88	90.0	91.2	-	0.88	91.7	92.6	-	0.88	89.1	92.3	-
0.72	65.1	78.0	1	0.86	91.4	93.2	-	0.86	87.9	90.7	-	0.86	90.2	92.7	-	0.86	85.4	89.8	1
0.68	65.7	72.9	1	0.84	89.2	91.2	-	0.84	85.8	89.7	-	0.84	88.1	91.1	-	0.84	85.2	88.5	-
0.64	63.2	68.0	1	0.82	92.6	93.1	-	0.82	89.0	90.4	-	0.82	89.3	91.1	-	0.82	80.1	83.9	1
0.6	60.9	65.5	-	0.8	90.4	90.9	-	0.8	86.8	89.2	-	0.8	87.1	89.9	-	0.8	78.8	81.9	1

difference between SpareLLM and GPT-4-Turbo is within 2% with a 95% confidence level. As our primary metric, we use cost savings, calculated as $\text{cost}_{\text{before}}/\text{cost}_{\text{after}}$. For example, if GPT-4-Turbo incurs a cost of \$300 and SpareLLM incurs a cost of \$30 on the same benchmark, this indicates a cost reduction by 10×. The accuracy metric is computed based on the ground-truth labels, while the equivalence metric is calculated by comparing the outputs to those of the reference LLM.

7.2 Experimental Results

Comparison against Baselines. Figure 3 shows the cost-accuracy tradeoff curves of SpareLLM, LLMCascade, and FrugalGPT across all benchmarks and LLM collections, with the x-axis on a logarithmic scale. Note that here we measure accuracy (against the ground-truth labels) instead of equivalence. For SpareLLM, we vary the equivalence constraint from 0.02 to 0.9. For LLMCascade, we set the consistency threshold from 0.4 to 1.0 as in [40]. We vary the cost budget for FrugalGPT from \$1 (i.e., effectively unbounded, as each input costs much less than a dollar) down to one-tenth of the cost of using the reference model. SpareLLM Pareto-dominates LLMCascade and FrugalGPT in most settings across all benchmarks, contributing to 91.1% and 83.8% of the points on the Pareto curve for OpenAI and Llama models, respectively. For the same accuracy level, SpareLLM provides cost savings of up to 3.4× and 7.4× compared to LLMCascade and FrugalGPT, respectively. SpareLLM selects the minimum-cost LLM that satisfies the given equivalence constraint by evaluating the difficulty of a specific task. In contrast, LLMCascade and FrugalGPT attempt to assess the difficulty of each individual instance. When dealing with a large number of instances, SpareLLM benefits from information gained during profiling, while the baselines incur redundant overheads across instances. In addition to offering a better cost-accuracy tradeoff, another key advantage of SpareLLM over LLMCascade and FrugalGPT is that it provides users with accuracy guarantees (based on Theorem 1). In contrast, it is unclear how a specific threshold value in LLMCascade translates to the final accuracy. Interestingly, for Llama models on the IMDB benchmark, the accuracy of FrugalGPT slightly decreases as the cost increases (at the end) because the Llama 3 70B model has higher accuracy than the Llama 3 405B model.

Runtime Overhead. Figure 4 depicts the runtime overhead of SpareLLM compared to the inference time of each LLM across all benchmarks, with the y-axis on a logarithmic scale. We compute the inference time of an LLM by summing the response times for all instances in the dataset. We measure the runtime overhead of SpareLLM by calculating its runtime excluding the inference time of the LLMs. The overhead of SpareLLM is relatively small, less than ten seconds, compared to the inference time of LLMs, which can extend to several hours.

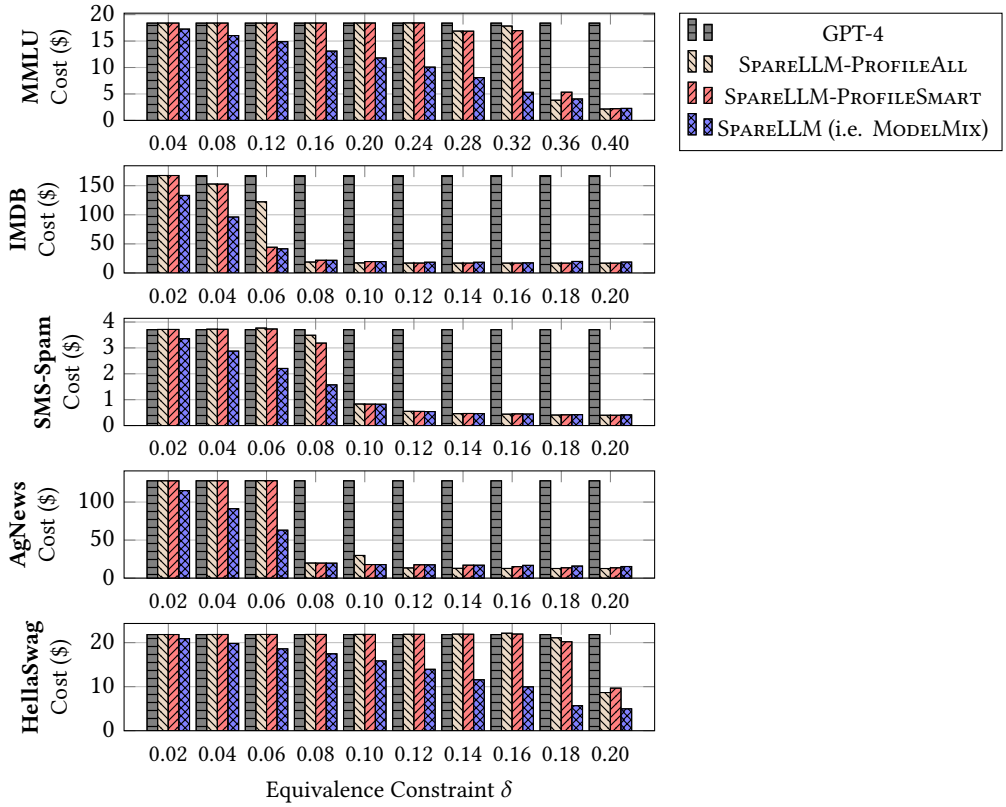


Fig. 5. Costs of GPT-4 and all variants of SpareLLM with varying equivalence constraint δ.

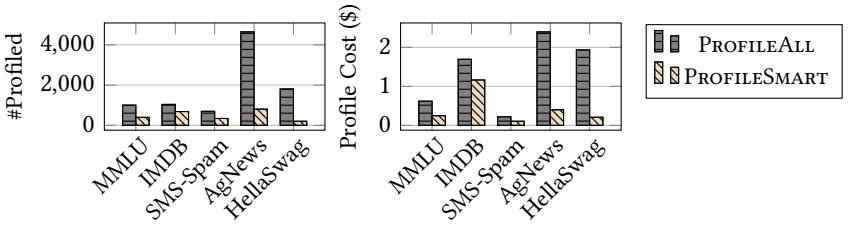


Fig. 6. Number of profiled instances (left) and profiling cost (right) for ProfileSmart compared to ProfileAll on all benchmarks.

$\langle\delta, \gamma\rangle$-Equivalence Guarantees. Table 3 presents the minimum and mean equivalence (%) of SpareLLM against the reference LLM across the ten runs for each benchmark and equivalence constraint. If the minimum equivalence meets or exceeds the equivalence threshold, SpareLLM produces outputs that satisfy the equivalence guarantees with respect to the reference model, GPT-4-Turbo, in all ten runs. Additionally, Table 3 details the number of runs, out of ten, in which the outputs of SpareLLM fail to meet the equivalence constraint. In most cases, the outputs of SpareLLM comply with the equivalence constraint for all ten runs. Across all benchmarks and equivalence constraints, merely 17 out of the 500 runs violate the equivalence constraint,

corresponding to 3.4% of cases. This rate is consistent with the confidence level $\gamma = 0.95$ used in our experiments.

Ablation Study. Figure 5 illustrates the costs associated with using GPT-4-Turbo and SpareLLM variants under varying user-defined equivalence constraints. We vary the equivalence constraint up to 0.4 for MMLU and up to 0.2 for IMDB, SMS-Spam, AgNews, and HellaSwag, dividing each range into ten equally spaced values. All SpareLLM variants largely outperform GPT-4-Turbo in terms of cost across all benchmarks. As the equivalence constraint δ is relaxed, SpareLLM achieves greater cost savings. With the strictest equivalence constraint, SpareLLM achieves cost savings of 1.1×, 1.3×, 1.1×, 1.1×, and 1.0× compared to GPT-4 for the MMLU, IMDB, SMS-Spam, AgNews, and HellaSwag benchmarks, respectively. When the equivalence constraint is increased to $\delta = 0.2$ for MMLU and $\delta = 0.1$ for IMDB, SMS-Spam, and AgNews, SpareLLM realizes greater savings of 1.6×, 8.6×, 4.5×, 7.2×, and 1.4×, respectively. This trend demonstrates the efficiency of SpareLLM in reducing costs as users impose less demanding equivalence requirements. We now discuss the variants of SpareLLM. With each variant of SpareLLM introducing a new feature to the preceding version, ProfileSmart improves upon ProfileAll by up to 2.8×, and ModelMix improves upon ProfileSmart by up to 3.2×. When the equivalence of an LLM is close to the equivalence threshold, ProfileAll struggles to conclusively accept or reject the LLM as satisfying the equivalence requirement. As a result, ProfileAll incurs additional overheads from continuous profiling without meeting the termination criterion. In contrast, ProfileSmart (and ModelMix) can terminate early if further profiling is expected to be wasteful based on cost estimations (see IMDB, $\delta = 0.6$). Figure 6 clearly show this improvement by comparing the number of profiled instances and the profiling cost of ProfileSmart with those of ProfileAll. Next, ModelMix outperforms ProfileSmart, particularly when the equivalence of all LLMs (except the reference LLM) is lower than or equal to the equivalence threshold (see IMDB, $\delta = 0.4$). Even when the more affordable LLMs fail to meet the equivalence constraint directly, ModelMix can leverage them in combination with the more accurate but more expensive reference LLM.

Breakdown per LLM. Figure 7 (left) provides a detailed view of SpareLLM in terms of the number of instances processed by each LLM. Here, the total number of instances processed by LLMs may exceed the number of instances in the dataset due to profiling, where the same item is processed using multiple LLMs. As the equivalence constraint δ is relaxed, SpareLLM seamlessly transitions to utilizing LLMs that, while less accurate, are more cost-efficient. For instance, when the equivalence constraint is stringent at $\delta = 0.02$ (0.04 for MMLU), GPT-4-Turbo is responsible for the majority of processing across all benchmarks. This aligns with expectations as the outputs of SpareLLM may differ from those of GPT-4 in no more than 2% of instances. On the other hand, with a more relaxed equivalence constraint of $\delta = 0.2$ (0.4 for MMLU), the processing portion of GPT-4-Turbo drops significantly to 1.4%, 0.1%, 0.7%, 0.1%, 9.9% for the MMLU, IMDB, SMS-Spam, AgNews, and HellaSwag benchmarks, respectively. A key advantage of SpareLLM is its ability to dynamically determine how many instances each LLM should process, while providing equivalence guarantees as per the user-specified equivalence constraint δ and confidence level γ. Figure 7 (right) also presents a breakdown of SpareLLM in terms of the cost per LLM. Although the number of processed items remains similar, the overall cost decreases as the equivalence constraint δ is relaxed, owing to the utilization of more cost-efficient LLMs.

Breakdown per Phase. Figure 8 presents a breakdown of SpareLLM with respect to the profiling and application phases. As before, we provide both the number of instances (left) and the inference cost (right). SpareLLM dynamically adjusts the number of instances to profile based on whether additional profiling is expected to benefit the application phase. For example, with $\delta = 0.1$ for SMS-Spam, SpareLLM profiles more instances compared to stricter equivalence constraints, which might initially seem to increase the cost. However, as illustrated in the figure, the overall cost

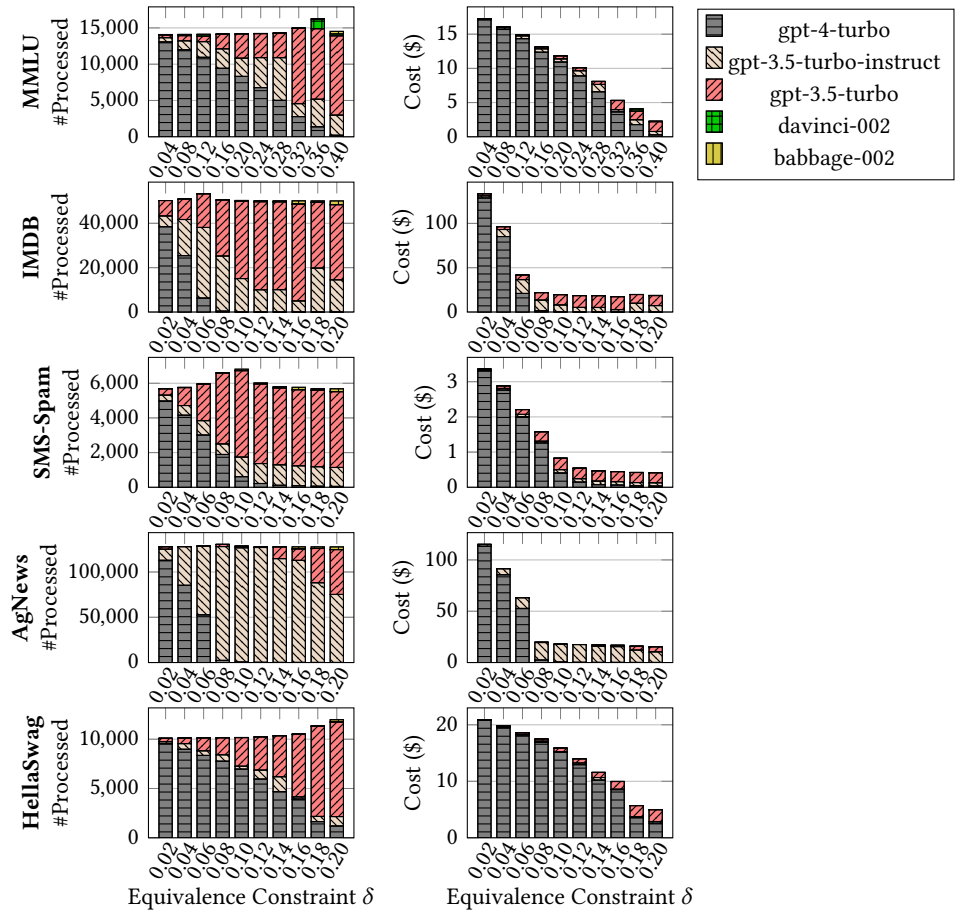


Fig. 7. Breakdown of SpareLLM in terms of the number of instances processed (left) and the cost (right) per LLM.

decreases because SpareLLM leverages this additional information during the application phase to utilize more cost-efficient LLMs.

Confidence Intervals on Equivalence. Figure 9 illustrates the lower and upper bounds of confidence intervals on the equivalence of each LLM during the profiling phase. We present the case where the equivalence threshold is set to $1 - \delta = 0.8$ for MMLU and $1 - \delta = 0.9$ for the other datasets, marked in red on the y-axis.As more items are profiled and additional information is collected, the confidence intervals narrow down further. If the upper confidence interval bound of an LLM falls below the equivalence threshold (as illustrated for babbage-002 on IMDB), the LLM is deemed `Invalid`, and this LLM is no longer profiled. Conversely, if the lower bound of the interval surpasses the equivalence threshold, the LLM is considered `Valid` (as demonstrated for gpt-3.5-turbo on IMDB). Profiling concludes upon identifying the most cost-efficient LLM with `Valid` status (for IMDB and SMS-Spam) when further profiling is expected to be wasteful (for MMLU, AgNews, and HellaSwag). The number of items processed for profiling is significantly less compared to the total number of input instances across all benchmarks.

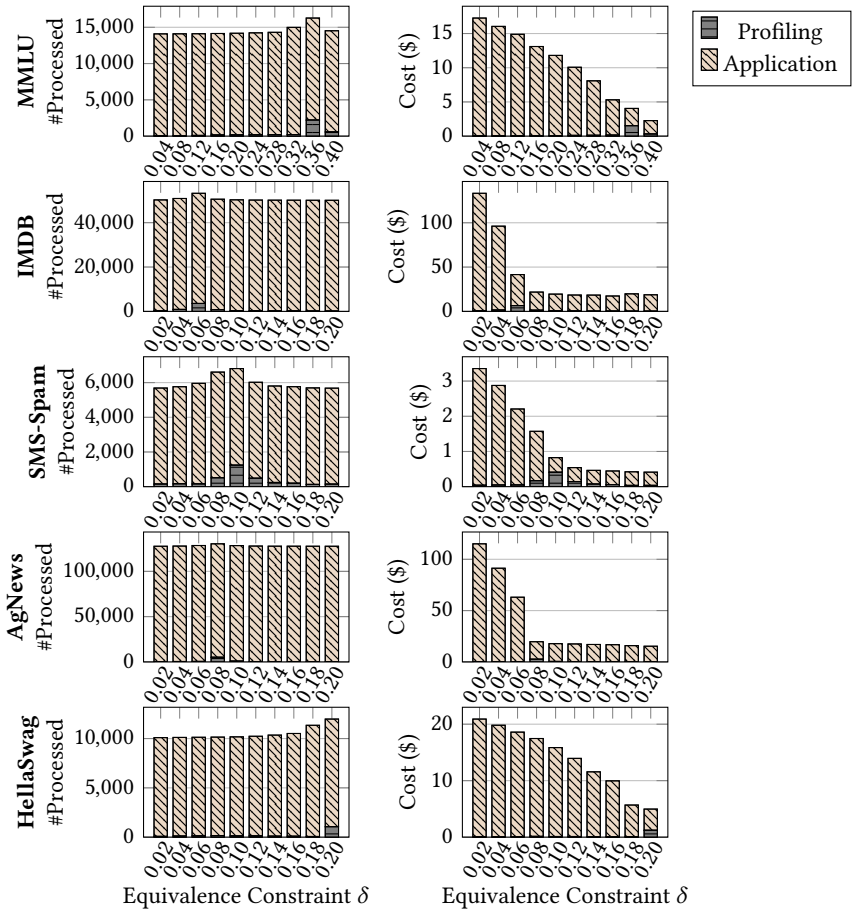


Fig. 8. Breakdown of SpareLLM in terms of the number of instances processed (left) and the cost (right) per phase.

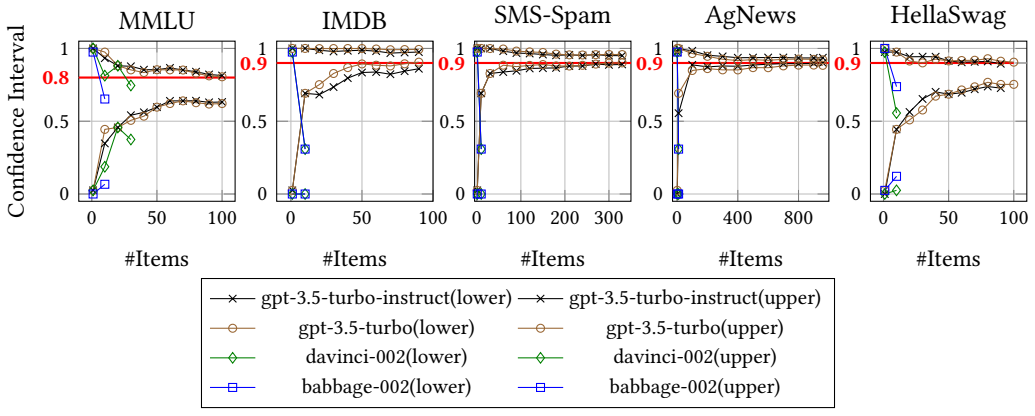


Fig. 9. Lower and upper bounds of confidence intervals on LLM equivalence during the profiling phase.

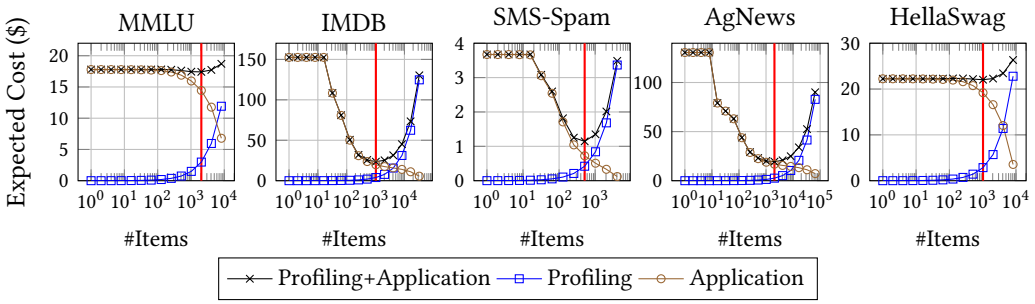


Fig. 10. Expected costs (including costs of the application phase) associated with profiling exactly k additional items with exponentially increasing k.

Cost Estimation during Profiling. Figure 10 illustrates the expected costs associated with profiling exactly k additional items. It is important to note that these expected costs include the impact of profiling on the anticipated costs of the subsequent application phase. Thus, we present the expected costs for both the profiling and application phases, as well as the total costs. SpareLLM enables the early termination of profiling, leveraging the cost estimations introduced in Section 5. As k increases exponentially, the expected costs display a clear pattern. As the number of profiled items increases, the expected cost for profiling also increases. Conversely, the expected cost of the application phase decreases as we gather more accurate information about the equivalence of LLMs. Thus, the overall expected cost exhibits the following pattern. Initially, increasing the number of profiled items reduces the associated expected costs until reaching a specific point. Beyond this point, further increases in the number of profiled items lead to increased expected costs. This pattern suggests the existence of an optimal k, at which the balance between the costs of profiling and the savings gained during the application phase is optimal. This optimal number k^* is marked with a red line in each plot. The benefits of profiling surpass its overheads up to a certain point, at which sufficient information about LLMs has been acquired. Beyond this point, profiling additional items yields little new information, making the costs of further profiling outweigh its benefits. This experiment features the estimated expected costs of profiling more items after SpareLLM has profiled 100 items. Here, we apply an equivalence constraint of $\delta = 0.2$ for MMLU and $\delta = 0.1$ for the other datasets.

Varying Input Difficulty. Figure 11 illustrates the number of instances processed (left) and the cost (right) of SpareLLM across varying input distributions. We divide the MMLU dataset into easy and hard examples based on whether gpt-3.5-turbo-1106 correctly answers the question. Then, we create new datasets, each containing 5,000 instances, with the ratio of hard inputs increasing from 0% to 50%. The equivalence constraint is set to $\delta = 0.2$. SpareLLM achieves greater cost savings when the dataset predominantly consists of easier instances. This is because SpareLLM can leverage cheaper models while satisfying the equivalence constraint. Interestingly, higher ratios of hard inputs do not necessarily lead to more profiling. Instead, profiling may terminate early when it becomes evident that cost-efficient LLMs are unlikely to meet the equivalence constraint.

8 Related Work

The Transformer architecture [36] revolutionized the field of natural language processing (NLP) with significant improvements across multiple tasks. Building on the Transformer architecture, a series of LLMs emerged, demonstrating state-of-the-art performance on various NLP tasks in both zero-shot [22] and few-shot settings [6, 37]. The number of parameters in LLMs has significantly increased along with their inference costs [8], as models with more parameters have shown

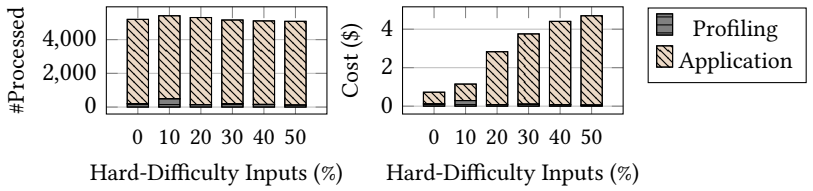


Fig. 11. Breakdown of SpareLLM across varying input difficulty levels, showing the number of instances processed (left) and the cost (right) per phase.

better performance [15, 20]. This has sparked interest in improving LLM inference efficiency [25], with strategies including cascade models [8, 17, 40, 42], query routing [11, 16, 28], early exit mechanisms [34, 35], caching results [44], batch prompting [9, 23], compressed LLMs [39], and distributed inference [38].

LLMCascade [40] is an LLM framework that uses a cascading pipeline of LLMs with varying costs, querying weaker LLMs multiple times to determine if a stronger LLM is needed. FrugalGPT [8] similarly employs LLM cascading but relies on ground truth labels to learn score thresholds for cost-efficient processing. Hybrid LLM [11] and RouteLLM [28] also require a labeled training set to train a router that selects the appropriate LLM for each input. In contrast, SpareLLM operates without ground truth labels and features a unique, principled approach to systematically determine how many items to process using the reference LLM. In general, model cascading and query routing frameworks reduce costs by adjusting compute per input complexity. On the other hand, SpareLLM targets scenarios where many input instances are evaluated for a common NLP task. Hence, SpareLLM models the difficulty of the NLP task and selects appropriate LLMs at the task level. Above all, SpareLLM provides equivalence guarantees on the generated outputs.

Instance-level profiling offers flexibility by dynamically adapting to varying input distributions. However, it introduces additional overhead, such as assessing the instance difficulty, detecting out-of-distribution inputs, and optimizing pipelines for each distribution, which often requires a labeled training set. In contrast, task-level profiling, as in SpareLLM, simplifies this process by identifying a cost-efficient LLM for the entire task through the evaluation of a fairly small number of instances. However, it assumes a relatively homogeneous distribution, which may limit its effectiveness in scenarios with highly diverse inputs. In the case of distribution shifts, SpareLLM can be improved to periodically verify the confidence intervals of LLMs to ensure equivalence guarantees.

SpareLLM is related to approximate query processing [7] in that it enables users to trade result precision for reduced costs. A significant body of prior work [1, 2, 18, 19, 26, 27, 31] allows users to specify an accuracy constraint in addition to their query. The IFocus algorithm [21] is relevant as it regularly updates confidence intervals to generate approximate visualizations of large datasets. Unlike these approaches that focus on relational data, SpareLLM is designed to process a large number of NLP task instances.

9 Conclusion

SpareLLM is a framework designed to minimize the cost of large-scale LLM inference for NLP tasks, while providing guarantees on the quality of its outputs. Through a novel profiling scheme and the strategic use of multiple LLMs, SpareLLM enables users to achieve significant cost savings safely, while ensuring high-quality results. Using OpenAI LLMs, our experimental evaluation across five real-world datasets demonstrates its ability to dramatically reduce costs compared to GPT-4-Turbo and LLM cascading baselines.

10 Acknowledgements

This material is based upon work supported by the National Science Foundation under Award No. 2239326.

References

[1] Sameer Agarwal, Barzan Mozafari, Aurojit Panda, Henry Milner, Samuel Madden, and Ion Stoica. 2013. BlinkDB: queries with bounded errors and bounded response times on very large data. In *Eighth Eurosys Conference 2013, EuroSys '13, Prague, Czech Republic, April 14-17, 2013*. ACM, 29–42. doi:10.1145/2465351.2465355

[2] Sameer Agarwal, Aurojit Panda, Barzan Mozafari, Anand P. Iyer, Samuel Madden, and Ion Stoica. 2012. Blink and It's Done: Interactive Queries on Very Large Data. *Proc. VLDB Endow.* 5, 12 (2012), 1902–1905.

[3] Together AI. 2025. Together AI - The AI Acceleration Acceleration Cloud. https://www.together.ai/.

[4] Tiago A. Almeida, José María Gómez Hidalgo, and Akebo Yamakami. 2011. Contributions to the study of SMS spam filtering: new collection and results. In *Proceedings of the 2011 ACM Symposium on Document Engineering, Mountain View, CA, USA, September 19-22, 2011*. ACM, 259–262.

[5] Lawrence D Brown, T Tony Cai, and Anirban DasGupta. 2001. Interval estimation for a binomial proportion. *Statistical science* 16, 2 (2001), 101–133.

[6] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

[7] Surajit Chaudhuri, Bolin Ding, and Srikanth Kandula. 2017. Approximate Query Processing: No Silver Bullet. In *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, USA, May 14-19, 2017*. ACM, 511–519.

[8] Lingjiao Chen, Matei Zaharia, and James Zou. 2023. FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance. *CoRR* abs/2305.05176 (2023).

[9] Zhoujun Cheng, Jungo Kasai, and Tao Yu. 2023. Batch Prompting: Efficient Inference with Large Language Model APIs. *CoRR* abs/2301.08721 (2023).

[10] Charles J Clopper and Egon S Pearson. 1934. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika* 26, 4 (1934), 404–413.

[11] Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. 2024. Hybrid LLM: Cost-Efficient and Quality-Aware Query Routing. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

[12] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. 2024. The Llama 3 Herd of Models. *CoRR* abs/2407.21783 (2024).

[13] Google. 2024. Gemini - Google DeepMind. https://deepmind.google/technologies/gemini/.

[14] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring Massive Multitask Language Understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

[15] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George

van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training Compute-Optimal Large Language Models. *CoRR* abs/2203.15556 (2022).

[16] Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. 2024. RouterBench: A Benchmark for Multi-LLM Routing System. *CoRR* abs/2403.12031 (2024).

[17] Wittawat Jitkrittum, Neha Gupta, Aditya Krishna Menon, Harikrishna Narasimhan, Ankit Singh Rawat, and Sanjiv Kumar. 2023. When Does Confidence-Based Cascade Deferral Suffice?. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

[18] Saehan Jo and Immanuel Trummer. 2020. BitGourmet: Deterministic Approximation via Optimized Bit Selection. In *10th Conference on Innovative Data Systems Research, CIDR 2020, Amsterdam, The Netherlands, January 12-15, 2020, Online Proceedings*. www.cidrdb.org.

[19] Srikanth Kandula, Anil Shanbhag, Aleksandar Vitorovic, Matthaios Olma, Robert Grandl, Surajit Chaudhuri, and Bolin Ding. 2016. Quickr: Lazily Approximating Complex AdHoc Queries in BigData Clusters. In *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*. ACM, 631–646.

[20] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling Laws for Neural Language Models. *CoRR* abs/2001.08361 (2020).

[21] Albert Kim, Eric Blais, Aditya G. Parameswaran, Piotr Indyk, Samuel Madden, and Ronitt Rubinfeld. 2015. Rapid Sampling for Visualizations with Ordering Guarantees. *Proc. VLDB Endow.* 8, 5 (2015), 521–532.

[22] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large Language Models are Zero-Shot Reasoners. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

[23] Jianzhe Lin, Maurice Diesendruck, Liang Du, and Robin Abraham. 2023. BatchPrompt: Accomplish more with less. *CoRR* abs/2309.00384 (2023).

[24] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning Word Vectors for Sentiment Analysis. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*. The Association for Computer Linguistics, 142–150.

[25] Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Hongyi Jin, Tianqi Chen, and Zhihao Jia. 2023. Towards Efficient Generative Large Language Model Serving: A Survey from Algorithms to Systems. *CoRR* abs/2312.15234 (2023).

[26] Supriya Nirkhiwale, Alin Dobra, and Christopher M. Jermaine. 2013. A Sampling Algebra for Aggregate Estimation. *Proc. VLDB Endow.* 6, 14 (2013), 1798–1809.

[27] Frank Olken and Doron Rotem. 1995. Random sampling from databases: a survey. *Statistics and Computing* 5 (1995), 25–42. doi:10.1007/BF00140664

[28] Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. 2024. RouteLLM: Learning to Route LLMs with Preference Data. *CoRR* abs/2406.18665 (2024).

[29] OpenAI. 2023. GPT-4 Technical Report. *CoRR* abs/2303.08774 (2023).

[30] OpenAI. 2024. OpenAI API. https://platform.openai.com/.

[31] Yongjoo Park, Barzan Mozafari, Joseph Sorenson, and Junhao Wang. 2018. VerdictDB: Universalizing Approximate Query Processing. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*. ACM, 1461–1476.

[32] Anthropic PBC. 2024. Anthropic. https://www.anthropic.com/.

[33] Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, H. Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant M. Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew J. Johnson, Blake A. Hechtman, Laura Weidinger, Iason Gabriel, William Isaac, Edward Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. 2021. Scaling Language Models: Methods, Analysis & Insights from Training Gopher. *CoRR* abs/2112.11446 (2021).

[34] Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. 2022. Confident Adaptive Language Modeling. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

[35] Roy Schwartz, Gabriel Stanovsky, Swabha Swayamdipta, Jesse Dodge, and Noah A. Smith. 2020. The Right Tool for the Job: Matching Model and Instance Complexities. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*. Association for Computational Linguistics, 6640–6651.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. 5998–6008.

[37] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

[38] Bingyang Wu, Yinmin Zhong, Zili Zhang, Gang Huang, Xuanzhe Liu, and Xin Jin. 2023. Fast Distributed Inference Serving for Large Language Models. *CoRR* abs/2305.05920 (2023).

[39] Zhaozhuo Xu, Zirui Liu, Beidi Chen, Yuxin Tang, Jue Wang, Kaixiong Zhou, Xia Hu, and Anshumali Shrivastava. 2023. Compress, Then Prompt: Improving Accuracy-Efficiency Trade-off of LLM Inference with Transferable Prompt. *CoRR* abs/2305.11186 (2023).

[40] Murong Yue, Jie Zhao, Min Zhang, Liang Du, and Ziyu Yao. 2024. Large Language Model Cascades with Mixture of Thought Representations for Cost-Efficient Reasoning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

[41] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a Machine Really Finish Your Sentence?. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, Anna Korhonen, David R. Traum, and Lluís Màrquez (Eds.). Association for Computational Linguistics, 4791–4800.

[42] Jieyu Zhang, Ranjay Krishna, Ahmed Hassan Awadallah, and Chi Wang. 2023. EcoAssistant: Using LLM Assistant More Affordably and Accurately. *CoRR* abs/2310.03046 (2023).

[43] Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level Convolutional Networks for Text Classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. 649–657.

[44] Banghua Zhu, Ying Sheng, Lianmin Zheng, Clark W. Barrett, Michael I. Jordan, and Jiantao Jiao. 2023. On Optimal Caching and Model Multiplexing for Large Model Inference. *CoRR* abs/2306.02003 (2023).

Received October 2024; revised January 2025; accepted February 2025