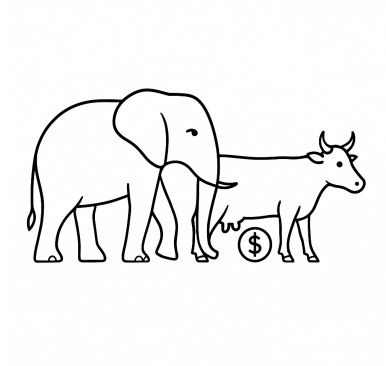
White Elephants and Cash Cows: Economically Wrangling the Zoo of AI Models

Thesis by
Michael J. Zellinger

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy



CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2026
Defended July 17, 2025

Michael J. Zellinger
ORCID: 0009-0001-7499-148X

ABSTRACT

The capabilities of artificial intelligence are rapidly expanding, but deploying AI systems in practice still poses significant challenges. Specifically, practitioners find limited guidance on selecting the most suitable AI model for a concrete use case, balancing the economics of an AI deployment, and managing the risk of AI errors. These challenges call for a unified framework addressing pain points in a conceptually clear and statistically sound manner. In this thesis, we present several components of such a framework: 1) uncertainty-aware system optimization, 2) economic evaluation, 3) error reduction with human-in-the-loop, and 4) a proof-of-concept system for synthetic data generation. Our work presents novel technical and conceptual approaches for orchestrating natural language-based systems, advancing the economical and reliable deployment of artificial intelligence.

PUBLISHED CONTENT AND CONTRIBUTIONS

Zellinger, Michael J. and Peter Bühlmann (2025). "Natural Language-Based Synthetic Data Generation for Cluster Analysis." In: *Journal of Classification*. DOI: 10.1007/s00357-025-09501-w. URL: https://doi.org/10.1007/s00357-025-09501-w.
MJZ conceived the project, wrote computer code, and prepared the manuscript, receiving valuable feedback and advice from PB.

Zellinger, Michael J. and Matt Thomson (2025a). "Economic Evaluation of LLMs." In: *arXiv preprint*. arXiv: 2507.03834 [cs.AI].
MJZ conceived the project, wrote computer code, and prepared the manuscript, receiving valuable feedback and advice from MT.

– (2025b). "Fail Fast, or Ask: Mitigating the Deficiencies of Reasoning LLMs with Human-in-the-Loop Systems Engineering." In: *arXiv preprint*. arXiv: 2507.14406 [cs.AI]. URL: https://arxiv.org/abs/2507.14406.
MJZ conceived the project, wrote computer code, and prepared the manuscript, receiving valuable feedback and advice from MT.

– (2025c). "Rational Tuning of LLM Cascades via Probabilistic Modeling." In: *Transactions on Machine Learning Research*. ISSN: 2835-8856. URL: https://openreview.net/forum?id=YCBVcGSZeR.
MJZ conceived the project, wrote computer code, and prepared the manuscript, receiving valuable feedback and advice from MT.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Chapter 1

INTRODUCTION

white elephant *noun*
a property requiring much care and expense and yielding little profit

cash cow *noun*
one regarded or exploited as a reliable source of money

1.1 Plan and Scope

Through the 2010s, AI researchers wondered by what time a machine could converse cogently and coherently enough to pass as human. This success criterion, known as the Turing test, was widely held as an intractable problem that would not give way in the foreseeable future. Then, in 2023, OpenAI's release of GPT-4 arguably shattered the Turing test. However, the world quietly moved on without fanfare. Too busy to pause in awe at this newly kindled Promethean fire, entrepreneurs and executives swiftly mobilized to seize the commercial opportunities presented by a vision of human intelligence dispensed by machines.

In the enterprise, unglamorous document-processing tasks soon took the spotlight. Could large language models (LLMs) make it easier to search across a business's vast internal corpus of unstructured documents? Would the models' programming abilities—including in database query languages such as SQL—enable non-technical employees to browse a company's internal data stores simply by speaking to them in English? Would the models' conversational abilities enable a full automation of customer support?

These visions are rapidly coming to fruition. For example, retrieval-augmented generation (Lewis et al., 2020) has become a popular technique for grounding AI answers in companies' internal document stores; text-to-SQL has become a mainstream offering among cloud data platforms; and AI customer service agents are proliferating.

However, the last mile of AI adoption is often the hardest. Practitioners still face many challenges when deploying LLMs to carry out real-world tasks, especially when those use cases fall outside "poster boy" offerings such as retrieval-augmented

generation, code generation, and customer support. From a high level point of view, practical pain points include selecting suitable AI models from a growing zoo of available options, balancing the economics of their usage, and managing the risk of AI errors.

Selecting Suitable AI Models. Following the success of generative AI, the zoo of AI models available to practitioners has greatly expanded. In addition to frontier LLMs developed by a handful of leading companies (Anthropic, OpenAI, xAI, Deepseek, Google, etc.), a large universe of open-source models is distributed to practitioners by cloud-based model providers such as Fireworks AI or Together AI. Practitioners often access both closed- and open-source models via software-only *application programming interfaces* (APIs), rather than invest in on-premises computing infrastructure.

This usage pattern makes switching between AI models as easy as changing a few lines of code, but it is typically not clear how to select the best model. The problem arises because model selection is a multi-objective problem: not only is the quality of a model's outputs of interest, but we would also like these outputs to be delivered fast and at reasonable cost. Naturally, slower and more expensive models tend to perform better, so practitioners must *decide* to settle for specific levels of quality—see Chapter 3 for an economic framework to guide this decision.

Balancing Economics. Outside of Silicon Valley, information technology (IT) has traditionally been viewed as a cost center. Computation is considered on the same footing as plumbing and electricity—an essential service whose reliability goes unquestioned except for conspicuous outages. AI models change the role of IT, shifting the job description of a computer from precisely executing minutely crafted sequences of steps to flexibly pursuing ambiguously defined goals. This shift carries important ramifications for the pricing of IT resources.

Language models "think" by emitting streams of tokens, and the number of the tokens produced determines both the duration and cost of a computation. This pricing model radically differs from traditional IT costs, which are purely operational—once a computer script has been written, executing it is only a matter of running it on suitable hardware powered by modest amounts of electricity. By contrast, deploying AI models implies paying for intelligence on demand. Starting with a blank slate, you may ask for the solution to a complex problem, and it shall be delivered to you out of nothingness. However, the bill rises in direct proportion to the required intellectual lift.

This pay-by-intelligence model puts LLMs into an intermediate position relative to human labor and traditional IT. For high-scale computation with large numbers of queries—say, a for-loop iterating through millions of database records—LLMs are much more expensive than the traditional approach of first writing and then executing a deterministic computer program. However, for tasks requiring nuanced understanding and logical insight, such as solving mathematics problems, an LLM's "wages" are much lower than a comparable human's. For example, as of June 2025, a state-of-the-art reasoning LLM charges \$0.01-\$0.10 for the solution of a competition-level high-school mathematics problem (Zellinger and Thomson, 2025a). In comparison, a human tutor hired at a rate of \$60/hour would ask for \$5 as compensation for five minutes of his time, a 50-100x increase in cost. See Chapter 3 for economic analysis along these lines.

Managing the Risk of AI Errors. Many practitioners are most concerned about the risk of AI models making mistakes. These fears are especially pronounced for executives in highly regulated industries with extensive compliance requirements, such as legal, medicine, and finance. Their concerns are well-founded: compared to standard computer code, which deterministically executes the same sequence of steps each time it is run, large language models generate their outputs *probabilistically*. The same input can result in different outputs, mimicking a characteristic of human work. The consequences can be innocuous, such as when an AI model responds to the question "What is the capital of Australia?" by saying either "Canberra is the capital of Australia" or "The capital of Australia is Canberra." However, the AI model may occasionally blurt out "Sydney"—an amusing blunder, perhaps, but not to the anxious executive who must take responsibility for this buffoonery.

Just as with human work, enhancing the reliability of work performed by AI models requires implementing checks and guardrails. In the first place, however, it is necessary to form a clear concept of what constitutes "right" vs "wrong" outputs. This binary assessment provides a basis for quantifying the probability that an AI model will make a mistake when processing a specific query. We will develop this notion further in Chapter 2. In Chapter 4, we explore deferring difficult queries to a human expert to further reduce AI error rates.

In summary, many challenges remain for effectively deploying AI models in practice. Among these pain points, we count 1) selecting suitable AI models, 2) balancing the economics of their usage, and 3) managing the risk of AI errors.

Presented with little reliable guidance, practitioners often seek refuge in social

media posts authored by self-styled "AI influencers" with dubious credibility or ulterior motives. We believe practitioners would benefit from a unified framework that addresses the challenges of practical LLM deployments in a conceptually clear, statistically sound, and empirically effective manner. This thesis presents a few components of such a framework:

- Uncertainty-Aware System Optimization (Chapter 2)
- Economic Evaluation (Chapter 3)
- Error Reduction with Human-in-the-Loop (Chapter 4)
- Proof-of-Concept System for Synthetic Data Generation (Chapter 5)

Before describing the chapters ahead in more detail, we give an overview over key concepts that will recur during our explorations.

1.2 Key Concepts

This section provides an overview and glossary of important concepts that will recur and reverberate throughout our explorations.

LLM. Large language models (LLM) are deep neural networks, commonly designed with a transformer architecture (Vaswani et al., 2017). The size of the model—measured in billions of parameters—largely determines its performance, with bigger models performing better (Kaplan et al., 2020).

Reasoning LLM. Traditional LLMs have shown weakness on answering numerical or logical questions, even when their parameter count is large (Brown et al., 2020b). To address this shortcoming, a special class of LLMs—termed *reasoning* models—is trained using reinforcement learning to think logically until a convincing answer has been identified (DeepSeek AI, 2025). Reasoning LLMs sharply outperform traditional LLMs on complex reasoning tasks, including mathematics. By automatically adapting their thinking duration to the difficulty of the query, these models apply a form of *test-time compute*—solving difficult problems by generating greater numbers of tokens at deployment time (Muennighoff et al., 2025).

Accuracy. The capability of LLMs is typically evaluated by testing them on queries with known correct answers. The *accuracy* is the fraction of correctly answered queries. More broadly, we use accuracy as a synonym for the quality of an LLM's

output. For example, in some applications, high accuracy takes the form of high scores on a continuous quality measure such as a grade from 0 to 100.

Cost. The *cost* of an LLM is the market price (in US dollars) for accessing and deploying LLMs via third-party APIs. This cost is typically billed per million tokens. For example, as of July 23rd, 2025, OpenAI charges $2 per million input tokens and $8 per million output tokens for usage of its flagship reasoning model, o3. Companies who invest in private physical infrastructure (server racks of graphics-processing units) face different economics, which are beyond the scope of this thesis.

Latency. The *latency* refers to the time interval—typically measured in seconds or minutes—that passes from when a query is sent to an AI model until the response is received. Although this period generally includes internet roundtrip delays, such factors are usually negligible compared to the primary source of latency: the AI model's internal computation time.

Uncertainty. Nobody is perfect: even state-of-the-art AI models occasionally make mistakes. We represent the *uncertainty* of a model by the probability that it will (in)correctly answer a given query. This uncertainty can be estimated in various ways, for example by following Kadavath et al. (2022) or computing the semantic entropy (Farquhar et al., 2024). Importantly, such probability estimates are only useful if they are *calibrated*.

Calibration. The *calibration* of a probability estimate signifies its agreement with the true underlying probability. For example, if we predict that an LLM will correctly answer a given query with 80% probability, the model should correctly answer such queries around 80% over time.

Confidence. Ideally, uncertainty quantification should be calibrated, but even uncalibrated metrics correlated with a model's uncertainty can be valuable. We refer to these metrics as *confidence*. In particular, confidence-gating—rejecting the lowest-confidence fraction (say, the bottom 10%) of queries—can substantially enhance the model's conditional accuracy on the queries it chooses to answer. Queries flagged as low-confidence can be set aside, escalated to a human expert, or redirected to a more reliable model (see Chapter 4).

1.3 The Road Ahead

The chapters ahead lay out several components of a framework for economically and reliably deploying large language models.

Chapter 2 — Uncertainty-Aware System Optimization. This chapter introduces LLM cascades, an architectural pattern for reducing the cost and latency of an LLM deployment. We present a probabilistic modeling approach for optimizing the crucial *deferral thresholds* using continuous optimization. Our methodology quantifies an AI model's risk of error and the interactions between different models' errors, yielding an uncertainty-aware framework for system optimization.

Chapter 3 — Economic Evaluation. This chapter tackles the problem of selecting *the* best-suited model for a concrete use case. We start by framing LLMs (and LLM systems) as reward-seeking agents, then formulate the search for an optimal model in terms of interpretable economic quantities: what is the economic cost of making a mistake? How much would you pay to lower per-query latency by 1 sec? How expensive is it to send intractable queries to human experts for review?

Chapter 4 — Error Reduction with Human-in-the-Loop. State-of-the-art reasoning LLMs are powerful problem solvers. Nonetheless, their error rates are still too high for many risk-sensitive tasks. This chapter presents a methodology for reducing AI mistakes by sending risky queries to a human expert. In addition, we address a secondary problem of large reasoning models. Like ponderous trucks floundering into the fast lane of California's Interstate 5, these models are often too slow not to provoke frustration. Can we make reasoning LLMs faster without compromising their performance? Read on to find out.

Chapter 5 — Proof-of-Concept System for Synthetic Data Generation. This chapter presents a practical LLM system addressing a real-world need. In statistical cluster analysis, researchers rely on synthetic data benchmarks to evaluate algorithmic advances. Constructing a diverse and interpretable set of evaluation scenarios is essential to ensure the generalizability of research findings. Unfortunately, crafting evaluation scenarios is a laborious task, often requiring manual tuning of low-level geometric parameters such as cluster centers and covariance matrices. To address this challenge, we present a *high-level* synthetic data generator with a natural language interface: researchers simply describe their desired evaluation scenarios in English, and the generator takes care of the rest.

Chapter 2

UNCERTAINTY-AWARE SYSTEM OPTIMIZATION

Zellinger, Michael J. and Matt Thomson (2025). "Rational Tuning of LLM Cascades via Probabilistic Modeling." In: *Transactions on Machine Learning Research*. ISSN: 2835-8856. URL: https://openreview.net/forum?id=YCBVcGSZeR.

Abstract: *Understanding the reliability of large language models (LLMs) has recently garnered significant attention. Given LLMs' propensity to hallucinate, as well as their high sensitivity to prompt design, it is already challenging to predict the performance of an individual LLM. However, the problem becomes more complex for compound LLM systems such as cascades, where in addition to each model's standalone performance, we must understand how the error rates of different models interact. In this paper, we present a probabilistic model for the joint performance distribution of a sequence of LLMs, which enables a framework for rationally tuning the confidence thresholds of a LLM cascade using continuous optimization. Compared to selecting confidence thresholds using Bayesian optimization, our parametric Markov-copula model yields more favorable error-cost trade-offs, improving the area under the error-cost curve by 4.3% on average for cascades with $k \geq 3$ models. In the low-sample regime with $n \leq 30$ training examples, the performance improvement widens to 10.2%, suggesting that our framework's inductive assumptions about the interactions between the error rates of different LLMs enhance sample efficiency. Overall, our Markov-copula model provides a rational basis for tuning LLM cascade performance and points to the potential of probabilistic methods in analyzing systems of LLMs.*

2.1 Introduction

As LLMs become workhorses of the modern computing stack, systems of LLMs have received significant attention (Zaharia et al., 2024; Chen et al., 2024b). These approaches make it possible to adapt computational spending to the performance requirements at the query or task level (Kag et al., 2023; Chen, Zaharia, and Zou, 2023), yielding significant gains in operational efficiency. These gains are achievable even when accessing LLMs entirely via black-box API calls, by switching between models of different capabilities.

However, moving from single LLMs to LLM systems introduces significant additional complexity. To find the system's optimal operating point, it is important to understand not just the performance of individual models but also the interactions between their error rates. For example, in a simple two-model LLM cascade in which a small model delegates difficult queries to a large model, the large model's error rate increases conditional on receiving a query, since the small model's confidence gating induces an adverse selection (Zellinger and Thomson, 2024).

In this paper, we present a parametric probabilistic model for the joint distribution of the calibrated confidences of a sequence of k LLMs, providing a rational basis for understanding the performance of LLM cascades. We focus on cascades whose constituent models are ordered by size, from smallest to largest. Our probabilistic model is based on a Markov factorization, leveraging the insight that LLMs similar in size are more predictive of each other's confidence. After using logistic regression to calibrate each LLM's confidence, we account for the pairwise interactions between subsequent LLMs' error rates using bivariate copulas, providing a data-efficient model of cascade performance that performs well with as few as $n \leq 30$ training examples across six benchmarks.

Our Markov-copula model makes it possible to tune the confidence thresholds of an LLM cascade using continuous optimization.

Compared to selecting these thresholds via Bayesian optimization, our Rational Tuning framework yields increasingly better error-cost trade-offs as cascade length grows. For cascades with $k \geq 3$ models, our method improves the area under the error-cost curve by 4.3% on average. Compared to high-resolution grid search, the improvement is 2.0%. At the same time, our algorithm significantly improves runtime scaling compared to grid search. For example, we reduce scaling with respect to the cascade length k from exponential to low-order polynomial, making it much faster to tune longer cascades consisting of $k \geq 5$ models.

Relative to the prior literature on LLM cascades, our main contributions are as follows:

- We propose a generative probabilistic model for the joint distribution of the calibrated confidences of a sequence of LLMs, based on a Markov factorization, copula modeling, and mixed discrete-continuous marginal distributions. We demonstrate that our model fits the empirical data well: on the test sets, we report average Cramér-von Mises statistics of $\sqrt{n}\text{CvM} = 0.006$ for the

copula models and $\sqrt{\text{CvM}} = 4\%$ for the mixed discrete-continuous marginal distributions.

- Building on our Markov-copula model, we develop an algorithm for tuning the confidence thresholds of an LLM cascade using continuous optimization, based on an analytic probabilistic model. We demonstrate that as cascade length grows, our method increasingly outperforms the error-cost trade-offs obtained with Bayesian optimization and high-resolution grid search baselines. In addition, relative to grid search our method significantly improves the computational complexity of finding optimal confidence thresholds, turning the dependencies on cascade length and the desired resolution of the error-cost curve from intractable and high-order polynomial into low-order polynomial and linear, respectively.

In addition, we present comprehensive evidence that simple hyperparameter-free feature transforms significantly improve the performance of calibrating LLM confidence with logistic regression (Zellinger and Thomson, 2024), demonstrating a 28.2% average reduction in expected calibration error across 10 LLMs and 6 benchmarks.

2.2 Background and Related Work

Language Models: given a predefined token vocabulary $\mathcal{V}$, a large language model (LLM) M defines an autoregressive probability distribution $t \sim p(\cdot|t_1, ..., t_n)$ for the next token $t \in \mathcal{V}$ given a sequence of tokens $(t_1, ..., t_n) \in \mathcal{V}^n$. In this work, we focus on the overall input-output behavior of the model M. We let x stand for the entire query consisting of tokens $(t_1, ..., t_n)$ and write $M(x)$ for the sequence of tokens $(t_{n+1}, t_{n+2}, ...)$ obtained when repeatedly sampling $t_{j+1} \sim P(\cdot|t_1, ..., t_j)$ for $j \geq n$ until encountering a stop token $t_\varnothing \in \mathcal{V}$.

Language Model Cascades: a length-k LLM cascade $C = M_1 \to ... \to M_k$ routes an incoming query x sequentially from model M_i to M_{i+1} based on confidence measures $\Phi_i = \Phi_i(x) \in [0, 1]$. When x reaches M_i, the cascade returns $M_i(x)$ if $\Phi_i(x) > \phi_i$, where $\phi_i \in (0, 1)$ is a confidence threshold for model M_i. Otherwise, C forwards the query x to the next model, M_{i+1}. Writing $C_{\text{i:k}}$ for the subcascade $M_i \to ... \to M_k$ consisting of the last $k - i + 1$ models, the output $C(x)$ of the overall

cascade is defined recursively as

$$C(x) = \begin{cases} M_1(x) & \text{if } \Phi_1(x) > \phi_1 \text{ or } |\mathrm{C}| = 1 \\ C_{2:\mathrm{k}}(x) & \text{otherwise,} \end{cases} \tag{2.1}$$

where $|C|$ is the length of the cascade, for example $|C_{2:k}| = k - 1$.

Different authors have recently explored LLM cascades. Chen, Zaharia, and Zou (2023) have shown that it is possible to approach the performance of a large LLM at much lower cost by initially sending queries to a small model; Aggarwal et al. (2024) present a flexible cascading approach based on a POMPD router; Yue et al. (2024) propose LLM cascades specifically for mathematical reasoning benchmarks; and Gupta et al. (2024) consider uncertainty at individual token position within longer generations. While many of these approaches use standard uncertain quantification techniques for LLMs (discussed below), some use trained neural networks for making the decision of forwarding a query x to the next model. Neural network approaches have the potential to make more finegrained distinctions between the capabilities of different LLMs[1], but may require large amounts ($n > 1000$) of task-specific training data to perform well.

Jitkrittum et al. (2024) discuss the limits of forwarding queries based purely on the confidence level of the current model, proposing to train a cascading decision that takes into account not only the current model's probability of correctness, but also that of the following model. In addition, Wang et al. (2024) explore finetuning LLMs to make them more effective as part of a cascade. Other methods for LLM orchestration use routers that directly forward queries to suitable LLMs in a one-to-many architecture (Ding et al., 2024; Kag et al., 2023; Sakota, Peyrard, and West, 2024; Hari and Thomson, 2023). In addition, some work has explored recombining the string outputs of several smaller models to yield improved performance (Jiang, Ren, and Lin, 2023).

Uncertainty Quantification and Calibration: LLMs within a cascades require the means to tell "easy" queries from "difficult" ones. Several authors have proposed methods for quantifying this uncertainty. These methods work in different ways. Some draw on the LLMs' intrinsic next-token probabilities (Hendrycks and Gimpel, 2018; Plaut, Nguyen, and Trinh, 2024), while others use prompting to elicit

[1] Of particular interest is the potential for detecting rare cases when a small model correctly answers a query on which a larger model fails.

confidence statements (Lin, Hilton, and Evans, 2022a; Kadavath et al., 2022; Xiong et al., 2024). Some sample repeatedly from the LLM and measure the consistency between different answers (Wang et al., 2023; Manakul, Liusie, and Gales, 2023; Farquhar et al., 2024; Lin, Trivedi, and Sun, 2024), while others train lightweight probes on top of an LLM's hidden embeddings (Azaria and Mitchell, 2023a; Ren et al., 2023a, Chen et al., 2024a; Kossen et al., 2024). Finally, it is even possible to evaluate uncertainty in a largely unsupervised way (Burns et al., 2024).

Calibration of LLM uncertainty refers to the question of whether numerical confidence scores reflect the true probabilities of error. Methods relying on LLMs' next-token probabilities face the challenge that these probabilities are typically overconfident, at least for instruction-tuned LLMs (Ouyang et al., 2022; OpenAI, 2024a). Although calibration is not required for forwarding queries based on confidence, it is important for accurately predicting error rates and desirable for gaining insights into system performance. Many techniques for calibration have been proposed (Platt, 1999; Zadrozny and Elkan, 2002; Naeini, Cooper, and Hauskrecht, 2015, Guo et al., 2017; Jiang et al., 2021). Temperature scaling, which divides an LLM's log probabilities by a suitable constant factor (typically >1), is often favored for its simplicity.

Copula Models: copula models are statistical tools for modeling joint probability distributions. They are widely used in applications. For example, in quantitative finance they are used to price complex securities such as baskets of loans whose repayments depend on multiple borrowers. Mathematically, a *copula* is a joint cumulative distribution function whose marginals all follow the uniform distribution. The idea behind copula modeling is that, in order to specify an arbitrary joint distribution $p(x, y)$, it suffices to specify the marginals $p(x)$, $p(y)$ along with a copula accounting for the correlation between x and y. This result is known as

Theorem 1 (Sklar's Theorem)**.** *Let F_{XY} be a joint distribution function with marginals F_X and F_Y. Then there exists a copula C such that for all $x, y \in \mathbb{R}$,*

$$F_{XY}(x, y) = C(F_X(x), F_Y(y)). \tag{2.2}$$

Conversely, if C is a copula and F_X and F_Y are distribution functions, then the distribution function F_{XY} defined by (2.2) is a joint distribution function with marginals F_X and F_Y.

For a proof and further discussion, see Nelsen (2006). Intuitively, copula modeling builds on the probability integral transform principle: if X is a continuous random variable with distribution function $F_X(\cdot)$, then $F_X(X)$ follows a uniform distribution (Casella and Berger, 2002). In our application to LLM cascades, we model the joint probability $p(\phi_{i-1}, \phi_i)$ of the calibrated confidences of models M_i and M_{i-1} using a Gumbel copula. This copula depends on a single correlation parameter θ, which can be easily calculated from the rank correlation (Kendall's τ) of the two variables.

2.3 Rational Tuning of LLM Cascades via Probabilistic Modeling

Markov-Copula Model

Our probabilistic model for the joint distribution of LLM confidences is based on calibrated confidence scores. We use logistic regression to transform a raw confidence signal p_{raw} into the calibrated confidence score

$$\phi = \Phi_\theta(p_{\text{raw}}), \tag{2.3}$$

where θ are the parameters of the logistic regression. The calibrated confidence ϕ estimates the model's probability of correctness based on the raw confidence signal p_{raw}. We calibrate each model separately, resulting in functions Φ_1, ..., Φ_k for the models $M_1, ..., M_k$ of a cascade $M_1 \to ... \to M_k$. See Section 2.4 for more details. Since the confidence signal p_{raw} depends on the query x, we also write $\phi = \Phi(x)$ in a slight abuse of notation.

Our probabilistic model for the joint distribution of the calibrated confidences $\Phi_1(x), ..., \Phi_k(x)$ consists of three parts. First, we model the marginal distribution of the calibrated confidence of each individual LLM in the cascade. Second, we model the correlation between the calibrated confidences $\Phi_i(x), \Phi_{i+1}(x)$ of adjacent models using copulas. Finally, we construct the full joint distribution by combining the conditional probabilities $p(\phi_{i+1}|\phi_i)$ using the Markov property.

Specifically, given a cascade $M_1 \to ... \to M_k$ with trained confidence calibrators $\Phi_1, ..., \Phi_k$, we first fit parametric univariate distributions $F_i(\phi_i|\theta_i)$ to model the true marginal distributions $\mathbb{P}(\Phi_i \leq \phi_i)$. Second, we account for the correlation between adjacent models by fitting copulas $C_{i,i+1}(\cdot, \cdot)$. Each copula $C_{i,i+1}$ makes it possible to compute the joint distribution $F_{i,i+1}(\cdot, \cdot)$ of (Φ_i, Φ_{i+1}) via

$$F_{i,i+1}(\phi_i, \phi_{i+1}) = C_{ij}(F_i(\phi_i), F_j(\phi_j)), \tag{2.4}$$

by Theorem 1. Finally, we estimate joint probabilities $\mathbb{P}(\Phi_1 \leq \phi_1, \Phi_2 \leq \phi_2, ..., \Phi_k \leq$

ϕ_k) by relying on the Markov assumption

$$\mathbb{P}(\Phi_i \leq \phi_i | \Phi_{i-1} \leq \phi_{i-1}, \Phi_{i-2} \leq \phi_{i-2}, ..., \Phi_1 \leq t_1) \approx \mathbb{P}(\Phi_i \leq \phi_i | \Phi_{i-1} \leq \phi_{i-1}), \tag{2.5}$$

which implies

$$\mathbb{P}(\Phi_1 \leq \phi_1, \Phi_2 \leq \phi_2, ..., \Phi_i \leq \phi_i) \approx \mathbb{P}(\Phi_1 \leq \phi_1) \prod_{j=2}^{i} \mathbb{P}(\Phi_j \leq \phi_j | \Phi_{j-1} \leq \phi_{j-1}), \tag{2.6}$$

for any $i = 2, ..., k$ and $\phi_1, ..., \phi_i \in (0, 1)$. We study the validity of assumption (2.5) in Section 2.4.

Parameter Inference for the Probabilistic Model

In this section, we describe in detail the components of our parametric probabilistic model and how we infer their parameters.

Continuous-discrete mixture of scaled beta distributions: to model the marginals of calibrated confidence, we must account for the possibility that LLMs sometimes return perfect confidence $p_{\text{raw}} = 1.0$, possibly as a result of performance optimizations such as quantization (Dettmers et al., 2024; Proskurina et al., 2024). Depending on the LLM and the task, almost half of all queries may return perfect confidence, as is the case of GPT-4o Mini on the MMLU validation set (45.7%).

To accommodate the resulting discrete probability masses at the minimum and maximum calibrated confidence values $\phi_{\min}$ and $\phi_{\max}$, we use a mixed continuous-discrete distribution based on a mixture of two beta distributions. Specifically, we use the distribution function

$$\begin{aligned} &F(\phi | w_1, w_2, \phi_{\min}, \phi_{\max}; \pi, \alpha_1, \beta_1, \alpha_2, \beta_2) \\ &\quad = w_{\min} \delta_{\phi_{\min}}(\phi) + w_{\max} \delta_{\phi_{\max}}(\phi) + (1 - w_{\min} - w_{\max}) F_{\text{mixture}}(\phi), \end{aligned} \tag{2.7}$$

where δ_z is the distribution of a point mass at z, and $F_{\text{mixture}}(\phi)$ is

$$\begin{aligned} &F_{\text{mixture}}(\phi | \phi_{\min}, \phi_{\max}; \alpha_1, \beta_1; \alpha_2, \beta_2) \\ &\quad = \pi F_\beta \left(\frac{\phi - \phi_{\min}}{\phi_{\max} - \phi_{\min}} \middle| \alpha_1, \beta_1 \right) + (1 - \pi) F_\beta \left(\frac{\phi - \phi_{\min}}{\phi_{\max} - \phi_{\min}} \middle| \alpha_2, \beta_2 \right). \end{aligned} \tag{2.8}$$

Here, $F_\beta(\cdot | \alpha, \beta)$ is the beta distribution with pdf $f_\beta(x | \alpha, \beta) = x^{\alpha-1}(1 - x)^{\beta-1}$ for $x \in (0, 1)$.

We infer the parameters of the model (2.7) as follows. First, we estimate the minimum and maximum calibrated confidences $\phi_{\min}$ and $\phi_{\max}$ by their observed minimum and maximum values on the training set. We estimate the corresponding discrete probability masses $w_{\min}$ and $w_{\max}$ by simple counting. Finally, to estimate the mixture of beta distributions (2.8), we use the expectation-maximization algorithm (Dempster, Laird, and Rubin, 1977).

Gumbel copula: to model the correlations between the calibrated confidences of pairs of LLMs, we use the Gumbel copula $C_\theta(u, v)$ given by

$$C_\theta(u, v) = \exp\left(-\left(\log\left(\frac{1}{u}\right)^\theta + \log\left(\frac{1}{v}\right)^\theta\right)^{\frac{1}{\theta}}\right), \tag{2.9}$$

where $\theta > 1$ measures the degree of correlation between u and v. To fit θ from empirical data, we use the relationship

$$\theta = \frac{1}{1-\tau}, \tag{2.10}$$

where τ is Kendall's rank correlation coefficient (Nelsen, 2006).

Tuning the Confidence Thresholds

The purpose of the Markov model (2.6) is to obtain optimal error-cost tradeoffs for an LLM cascade C by tuning the confidence thresholds. We formulate the optimization problem

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \left(1 - \mathbb{P}_{\boldsymbol{\theta}}(\text{Correct})\right) + \lambda\, \mathbb{E}_{\boldsymbol{\theta}}[\text{Cost}], \tag{2.11}$$

where $\boldsymbol{\theta} \in \mathbb{R}^{k-1}$ denotes the confidence thresholds $(\phi_1, ..., \phi_{k-1})$. The Lagrange multiplier $\lambda \geq 0$ indicates the user's cost sensitivity. Setting $\lambda = 0$ means that cost is irrelevant, whereas $\lambda > 0$ penalizes the use of expensive models. To compute the efficient frontier of optimal $(\mathbb{P}(\text{Correct}), \mathbb{E}[\text{Cost}])$ tuples, we solve (2.11) for different values of the cost sensitivity λ.

Since λ has no known relationship with the expected cost, it is not clear how to choose λ to obtain uniform coverage of the efficient frontier. In practice, we start with very small values of λ and set

$$\lambda \leftarrow (1 + r)\lambda, \tag{2.12}$$

for some $r > 0$, until the cost constraint is stringent enough to make the expected cost equal to the least expensive model's expected cost. Typically, setting r between 0.25

and 1 performs well. For any potential gaps in coverage, we adaptively interpolate the optimal thresholds. Specifically, if $\lambda^{(i)} < \lambda^{(i+1)}$ yield optimal thresholds $\boldsymbol{\theta}^{(i)}$ and $\boldsymbol{\theta}^{(i+1)}$ and the gap $|\boldsymbol{\theta}_j^{(i+1)} - \boldsymbol{\theta}_j^{(i)}| = |\phi_j^{(i+1)} - \phi_j^{(i)}|$ for any individual threshold exceeds probability mass q based on the distribution of the calibrated confidence Φ_j, we insert

$$\boldsymbol{\theta}^{(i+1/2)} = (\boldsymbol{\theta}^{(i)} + \boldsymbol{\theta}^{(i+1)})/2 \tag{2.13}$$

into the list of optimal thresholds between $\boldsymbol{\theta}^{(i)}$ and $\boldsymbol{\theta}^{(i+1)}$. We repeat the infilling procedure (2.13) until no gaps remain at level q. We have found $q < 0.2$ to perform well.

Efficient computation and optimization of the objective: solving the minimization problem (2.11) requires computing a cascade's probability of correctness and expected cost for candidate confidence thresholds $\boldsymbol{\theta} = (\phi_1, ..., \phi_{k-1}) \in \mathbb{R}^{k-1}$. To compute these quantities, we rely on the decompositions (2.14) and (2.15) presented in

Proposition 2. *Consider a cascade $M_1 \to ... \to M_k$ with confidence thresholds $(\phi_1, ..., \phi_{k-1})$. Assume that the distribution functions for the calibrated confidences Φ_i satisfy (2.5), for $i = 1, 2, ..., k$. Assume further that the expected numbers of input and output tokens, $T_i^{(in)}$ and $T_i^{(out)}$, for each model i are independent of the calibrated confidences $\Phi_1, ..., \Phi_k$. Then the probability of correctness $\mathbb{P}(Correct)$ and expected cost $\mathbb{E}[Cost]$ for the cascade are*

$$\begin{aligned}\mathbb{P}(Correct) = &\int_{\{\Phi_1 > \phi_1\}} \Phi_1(\omega)\, \mathrm{d}\mathbb{P}(\omega) \\ &+ \sum_{i=2}^{k} \mathbb{P}(\Phi_1 \le \phi_1) \left(\prod_{j=2}^{i-1} \mathbb{P}(\Phi_j \le \phi_j | \Phi_{j-1} \le \phi_{j-1}) \right) \\ &\times \int_{\{\Phi_i > \phi_i\}} \Phi_i(\omega)\, \mathrm{d}\mathbb{P}(\omega | \Phi_{i-1} \le \phi_{i-1})\end{aligned} \tag{2.14}$$

$$\begin{aligned}\mathbb{E}[Cost] = &(1 - \mathbb{P}(\Phi_1 \le \phi_1))\, \mathbb{E}[C_1] \\ &+ \sum_{i=2}^{k} \mathbb{P}(\Phi_1 \le \phi_1) \left(\prod_{j=2}^{i-1} \mathbb{P}(\Phi_j \le \phi_j | \Phi_{j-1} \le \phi_{j-1}) \right) \\ &\times (1 - \mathbb{P}(\Phi_i \le \phi_i | \Phi_{i-1} \le \phi_{i-1})) \sum_{j=1}^{i} \mathbb{E}[C_j]\end{aligned} \tag{2.15}$$

where C_i is the cost per query of model i. Specifically, if $\gamma_i^{(in)}$ and $\gamma_i^{(out)}$ are the costs per input and output token, $C_i = \gamma_i^{(in)} T_i^{(in)} + \gamma_i^{(out)} T_i^{(out)}$. To simplify the notation, we

let $\phi_k := -\infty$ *(although there is no confidence threshold for the final model in the cascade).*

Proof. See Appendix 2A for a proof. □

By leveraging the structure of the summands in Proposition 2, we efficiently compute (2.14) and (2.15) in $O(k)$ time, where k is the length of the cascade. See Appendix 22B for the algorithm. To solve the minimization problem (2.11), we use the L-BFGS-B optimizer, a low-memory version of the Broyden–Fletcher–Goldfarb–Shanno algorithm (Liu and Nocedal, 1989) modified to handle simple box constraints.

Smoothness of the objective: although our Markov-copula model uses mixed discrete-continuous marginals, the objective (2.11) is smooth because we restrict each threshold ϕ to vary only inside the interior of the interval $(\phi_{\min}, \phi_{\max})$, where the marginal distributions of calibrated confidence are smooth. Leaving out the boundary $\{\phi_{\min}, \phi_{\max}\}$ results in no loss of generality because selecting $\phi \in \{\phi_{min}, \phi_{max}\}$ is equivalent to dropping the model from the cascade (if $\phi = \phi_{\max}$) or dropping all subsequent models (if $\phi = \phi_{\min}$). Within our framework, it is possible to carry out such model selection by evaluating subcascades. After fitting copula models for all pairs of models (rather than only adjacent pairs), evaluating subcascades involves little computational overhead.

2.4 Results

Methodology

Forwarding Queries: the models in our cascades decide whether to forward queries by thresholding the calibrated confidence $\phi = \Phi(p_{\text{raw}})$, where p_{raw} is the raw confidence signal. We obtain p_{raw} from the model-intrinsic next-token probabilities. On multiple-choice tasks, we take the maximum probability among the answer choices (Hendrycks and Gimpel, 2018; Plaut, Nguyen, and Trinh, 2024). In the natural language generation case, we first generate the answer, then send a follow up verification prompt to the model asking "Is the proposed answer `<answer>` true? Answer only `Y` or `N`." We use the probability of the `Y` token as the confidence signal p_{raw}. Our prompt templates are available in Appendix 2C.

Since we focus on providing techniques compatible with black-box LLM inference via third-party APIs, we leave consideration of hidden layer-based confidence signals to future work. In addition, we do not consider resampling methods such as semantic

entropy (Farquhar et al., 2024). Such methods are compatible with black-box inference, but in the context of LLM cascades, their computational overhead appears prohibitive. For example, at the time of writing inference of Llama3.1 405B typically costs 15 times more than inference of Llama3.1 8B. In this case, it is likely preferable to directly run the 405B model once rather than forward a query based on $k \approx 15$ resamples of the 8B model. See Appendix 2D for a table listing small-large model pairings from Meta, Anthropic, and OpenAI, along with their price differentials.

Confidence Calibration: raw token probabilities of instruction-tuned LLMs are typically poorly calibrated (Ouyang et al., 2022; Brown et al., 2020b; OpenAI, 2024a; Plaut, Nguyen, and Trinh, 2024). However, calibration is important for accurate error prediction. To obtain calibrated confidence scores, we use logistic regression. We favor this approach over temperature scaling since it yields p values and other statistical metrics that are useful for diagnosing calibration issues, especially in a low-data scenario.

Unfortunately, the overconfidence of the raw token probabilities makes the distribution of raw confidence signals highly peaked. The raw token probabilities accumulate near 1.0, making tiny changes in confidence (for example, $p_{\text{raw}} = 0.98$ vs $p_{\text{raw}} = 0.99$) highly consequential. To enhance the calibration performance of logistic regression, as a pre-processing step we apply hyperparameter-free feature transformations that spread out the overconfident probabilities via asymptotes near $p_{\text{raw}} = 0.0$ and $p_{\text{raw}} = 1.0$. Following Zellinger and Thomson (2024), on multiple-choice tasks we use the transformation

$$\xi(p_{\text{raw}}) = \log\left(\frac{1}{1 - p_{\text{raw}}}\right), \tag{2.16}$$

whereas on natural language generation tasks, we use

$$\xi(p_{\text{raw}}) = \begin{cases} \log\left(\frac{1}{1-p_{\text{raw}}}\right) & \text{if } p \geq \frac{1}{2}, \\ \log\left(\frac{1}{p_{\text{raw}}}\right) & \text{if } p < \frac{1}{2}. \end{cases} \tag{2.17}$$

Importantly, these feature transformations do not require any hyperparameter tuning.

Unfortunately, models sometimes return perfect certainty $p_{\text{raw}} = 1.0$ or $p_{\text{raw}} = 0.0$, making (2.16) and (2.17) blow up. To address this problem, we reassign all observations with infinite ξ to the maximum of the finite values of ξ. In other words, we define

$$\xi_{\max} = \max\{\xi(p_{\text{raw}}) : (p_{\text{raw}}, y) \in \mathcal{D}, p_{\text{raw}} < \infty\}, \tag{2.18}$$

where $\mathcal{D}$ is the training set consisting of pairs (p_{raw}, y), and $y \in \{0, 1\}$ indicates correctness of the model's answer.[2] We set all observations where $\xi = \infty$ to $\xi_{\max}$, and treat $\xi_{\min}$ analogously.

Benchmarks: we evaluate our probabilistic model and the error-cost curves of LLM cascades on six language modeling benchmarks including MMLU (Hendrycks et al., 2021a); MedMCQA (Pal, Umapathi, and Sankarasubbu, 2022); TriviaQA (Joshi et al., 2017); XSum (Narayan, Cohen, and Lapata, 2018); GSM8K (Cobbe et al., 2021); and TruthfulQA (Lin, Hilton, and Evans, 2022b). These tasks include general-purpose knowledge and reasoning, domain-specific QA, open-domain QA, summarization, mathematical reasoning, and truthfulness in the face of adversarially chosen questions.

For each benchmark, we use 300 examples for training, and 1000 examples for testing, except on MMLU and TruthfulQA. On MMLU, the dev set contains only 285 examples, of which we use all. The validation set consists of 1531 examples and is divided into different subjects; to avoid bias from subject selection, we take all 1531 validation examples for testing. On TruthfulQA, the entire data set consists only of 817 observations, of which we randomly select 300 for training and the remaining 517 for testing.

Importantly, we run each benchmark in a **zero-shot** manner, since we believe this setting faithfully reflects off-the-shelf use of LLMs in practice. Appendix 2C gives the prompt templates we used for each benchmark. To conveniently transform and calibrate the raw confidence scores, track the numbers of input and output tokens, and monitor cost, we ran our evaluations using a preliminary version of the `niagara` Python package for LLM cascading. Code for reproducing the results of the paper is available on GitHub.[3].

Evaluation: to evaluate whether a model's answer is correct on open-ended questions, we use Anthropic's Claude 3.5 Sonnet model as a judge. Note that this judging task is relatively easy since the open-ended benchmarks provide reference answers. For example, on TruthfulQA, we include in the evaluation prompt for Claude a list of correct and incorrect reference answers, as provided by the authors of the benchmark (Lin, Hilton, and Evans, 2022b). On XSum, we do not use the one-

[2]Note we do not require knowing the *actual* answer of a model, only whether it was correct.

[3]Code for reproducing the results of the paper is available at `github.com/mzelling/rational-llm-cascades`.

Table 2.1: Overall performance of language models across tasks, evaluated on the $n \approx 1000$ test sets. %Corr is the percentage of correct answers, %ECE is the expected calibration error (after training on the $n \approx 300$ training sets), and %Cert is the percentage of queries for which a model returns log probabilities indicating certainty ($-\infty$ or 0.0).

	MMLU			MedMCQA			TriviaQA			XSum			GSM8K			TruthfulQA		
Model	**%Corr**	**%ECE**	**%Cert**	**%Corr**	**%ECE**	**%Cert**	**%Corr**	**%ECE**	**%Cert**	**%Corr**	**%ECE**	**%Cert**	**%Corr**	**%ECE**	**%Cert**	**%Corr**	**%ECE**	**%Cert**
llama3.2-1b	42.5	3.8	0.0	34.5	8.7	0.0	37.2	5.8	0.0	9.4	2.9	0.0	45.9	13.1	0.0	35.8	4.3	0.0
llama3.2-3b	57.2	4.0	0.0	53.1	6.8	0.0	63.3	4.5	0.0	21.2	3.6	0.0	79.2	9.5	0.0	43.3	7.5	0.0
llama3.1-8b	63.4	4.1	0.0	51.8	9.4	0.0	78.7	6.2	0.0	50.8	3.5	0.0	84.3	4.7	0.0	50.3	7.3	0.0
llama3.1-70b	81.5	**2.4**	0.0	72.6	9.9	0.0	92.8	2.3	0.0	84.5	6.0	0.0	94.9	2.9	0.0	59.4	5.7	0.0
llama3.1-405b	**85.2**	2.9	0.1	75.7	10.8	0.0	94.9	3.0	0.1	83.9	5.4	0.0	**97.1**	1.9	0.5	69.2	5.6	0.0
qwen2.5-32b-c	75.3	5.3	0.0	55.9	6.2	0.0	70.2	8.9	0.0	69.3	4.3	0.0	95.1	3.2	0.0	57.4	5.9	0.0
qwen2.5-72b	82.0	4.9	0.0	69.1	7.0	0.0	87.6	3.2	0.3	95.2	2.2	15.5	95.4	**1.2**	79.4	57.8	7.7	0.6
gpt-4o-mini	74.9	4.7	45.7	66.0	5.3	27.8	90.0	**2.8**	76.2	97.6	2.6	38.1	92.9	3.5	48.3	59.4	7.0	26.7
gpt-4o	83.6	4.8	22.7	**76.5**	**2.8**	4.8	**96.2**	2.1	0.9	**99.0**	**0.7**	0.0	95.9	2.1	4.3	**72.1**	**3.8**	0.2
Average	71.7	4.1	7.6	61.7	7.4	3.6	79.0	4.2	8.3	67.9	3.5	6.0	86.7	4.7	14.7	56.1	5.9	3.0

line reference summaries and instead follow G-Eval (Liu et al., 2023) to evaluate a proposed summary in terms of its coherence, consistency, fluency, and relevance (Kryściński et al., 2019). We ask Claude to score each dimension on a scale of 1-5. We consider a summary to be correct if it attains a perfect score (5) in each dimension.

Language Models: we work with models from Meta's Llama3 series (1B-405B), Alibaba's Qwen series (Qwen2.5 32B Coder and Qwen 72B), and OpenAI's GPT models (GPT-4o Mini and GPT-4o). All models are instruction-tuned. We used the OpenAI API to run inference with GPT-4o Mini and GPT-4o, and the Fireworks API for all other models.

Performance Summary

Tables 2.1 and 2.2 show the overall performance of all the language models across tasks, including the calibration performance. We measure calibration in terms of the expected calibration error (ECE), which we compute adaptively by bucketing confidence scores into 10 bins based on the deciles of their distributions. Tables 2.1 and 2.2 yield several interesting findings.

First, some of the models often return raw log probabilities indicating certainty ($-\infty$ or 1.0). This tendency varies strongly by model family. OpenAI's GPT models are especially prone to certainty: on MMLU, for example, GPT-4o Mini returns raw confidence 1.0 on 45.7% of queries, while GPT-4o does so on 22.7% of queries. By contrast, Llama3.1 405B returns perfect confidence only on 0.1% of queries.

Second, the test ECE for our calibration scheme varies by model and by benchmark. The benchmark yielding the poorest calibration is MedMCQA, giving an average test ECE of 7.4% across models. However, some models give exceptional calibration

Table 2.2: Expected calibration error for logistic regression-based calibration, with (%ECE) and without (%$\text{ECE}_{\text{-TF}}$) applying the nonlinear transformations (2.16) and (2.17) as a pre-processing step. All values are computed on the $n \approx 1000$ test sets, after fitting the logistic regressions on the $n \approx 300$ training sets. For each benchmark, bold font indicates the better performance. The column %Δ shows the reduction in ECE when using the transformations.

	MMLU			MedMCQA			TriviaQA			XSum			GSM8K			TruthfulQA		
Model	**%ECE**	**%$\text{ECE}_{\text{-TF}}$**	**%Δ**	**%ECE**	**%$\text{ECE}_{\text{-TF}}$**	**%Δ**	**%ECE**	**%$\text{ECE}_{\text{-TF}}$**	**%Δ**	**%ECE**	**%$\text{ECE}_{\text{-TF}}$**	**%Δ**	**%ECE**	**%$\text{ECE}_{\text{-TF}}$**	**%Δ**	**%ECE**	**%$\text{ECE}_{\text{-TF}}$**	**%Δ**
llama3.2-1b	**3.8**	6.1	-37.7	**8.7**	9.8	-11.2	5.8	5.8	0.0	2.9	**2.7**	7.4	**13.1**	13.2	-0.8	4.3	4.3	0.0
llama3.2-3b	**4.0**	7.4	-45.9	**6.8**	10.0	-32.0	**4.5**	14.9	-69.8	**3.6**	3.7	-2.7	9.5	9.5	0.0	**7.5**	6.4	17.2
llama3.1-8b	**4.1**	7.0	-41.4	**9.4**	14.1	-33.1	**6.2**	15.1	-58.9	**3.5**	9.7	-63.9	**4.7**	5.1	-7.8	7.3	7.3	0.0
llama3.1-70b	**2.4**	7.9	-69.6	**9.9**	12.5	-20.8	**2.3**	5.1	-54.9	**6.0**	10.4	-42.3	**2.9**	4.5	-35.6	5.7	**5.3**	7.5
llama3.1-405b	**2.9**	10.4	-72.1	**10.8**	14.2	-24.0	**3.0**	4.9	-38.8	**5.4**	10.2	-47.1	**1.9**	3.6	-47.2	**5.6**	10.8	-48.1
qwen2.5-32b-c	**5.3**	13.9	-61.9	**6.2**	14.5	-57.2	**10.0**	15.7	-36.3	**4.3**	10.2	-57.8	**3.2**	4.7	-31.9	**5.9**	10.6	-44.3
qwen2.5-72b	**4.9**	10.9	-55.0	**7.0**	16.1	-56.5	**4.0**	9.4	-57.4	**2.2**	3.2	-31.3	**1.2**	4.9	-75.5	**7.5**	9.3	-19.4
gpt-4o-mini	**4.7**	14.8	-68.2	**5.3**	15.5	-65.8	**1.4**	3.8	-63.2	2.6	**2.4**	8.3	**3.5**	6.1	-42.6	**5.3**	5.8	-8.6
gpt-4o	**4.8**	11.2	-57.1	**3.1**	12.6	-75.4	**2.1**	3.6	-41.7	**0.7**	1.0	-29.6	**2.1**	4.6	-54.3	**3.8**	11.4	-66.7
Average	5.0	7.9	-36.7	7.6	9.8	-22.5	4.5	8.2	-45.0	3.1	4.3	-28.2	6.2	7.1	-12.7	5.1	6.7	-23.9

performance across benchmarks. GPT-4o stands out: its test ECE never exceeds 4.8%, which is its ECE on MMLU.

Overall, we observe that our calibration scheme performs satisfactorily across benchmarks and models, with most benchmarks reporting an average test ECE below 5%. Table 2.2 ablates the importance of the hyperparameter-free feature transforms (2.16) and (2.17) for obtaining effective calibration. Applying these transformations results in much lower test ECE scores, reducing them by 28.2% on average. Figure .1 in Appendix 2E further verifies calibration by showing that, across models and benchmarks, rejecting queries for which the calibrated confidence is $< 1 - q$ approximately lowers the test error rates to $< q$.

Goodness-of-Fit of the Markov-Copula Model

In this section, we show that our probabilistic model fits the empirical data well. We start by presenting evidence that the Markov assumption (2.5) approximately holds. Second, we show that our Gumbel copula models successfully account for correlations between the error rates of different LLMs, as measured by low square-rooted Cramér-von Mises (CvM) statistics and low rejection rates of the null hypothesis. Finally, we show that our mixed discrete-continuous mixtures of beta distributions provide an adequate model for the marginal confidence distributions, as measured by low square-rooted CvM scores. However, the high rejection rates of the null hypothesis suggest the potential for further improvements.

Verifying the Markov Assumption

To verify that (2.5) approximately holds, we first visualize the rank correlation between the calibrated confidences of different models. Figure 2.1 shows that the

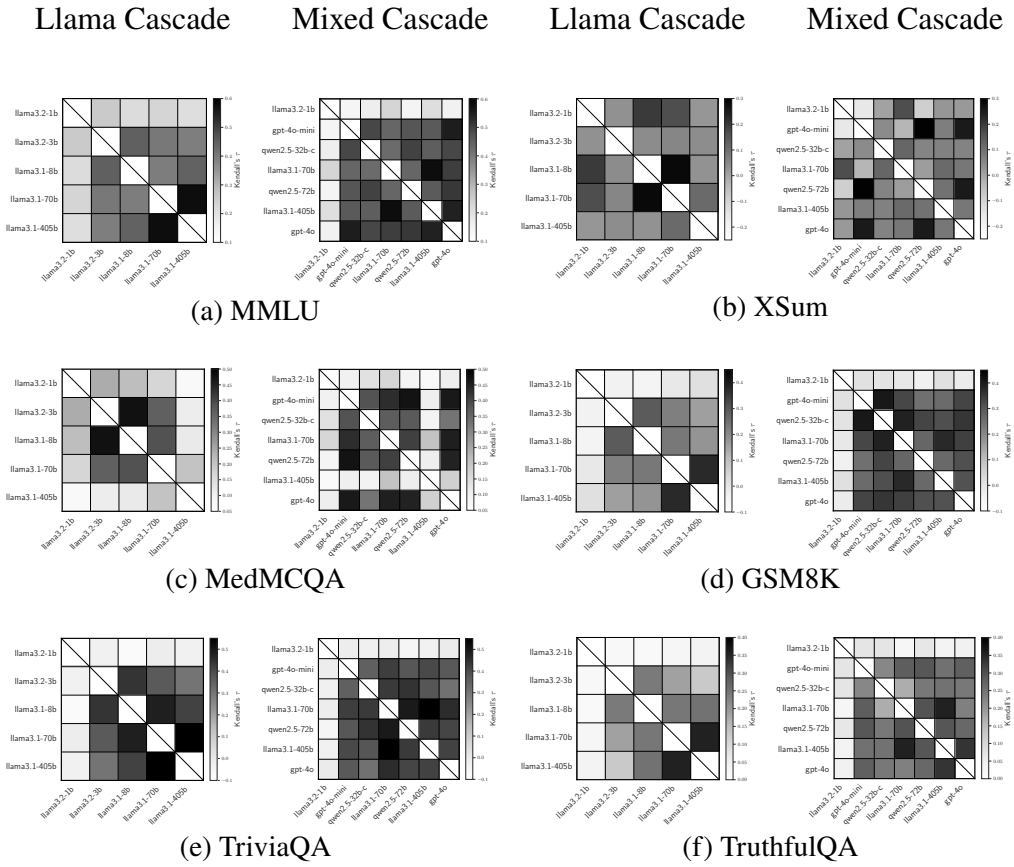


Figure 2.1: Evaluates the Markov property by showing the Kendall's τ rank correlation between the calibrated confidences of pairs of LLMs, as evaluated on the test set ($n \approx 1000$ examples). In a Markov pattern, the largest rank correlations occur near the diagonal, based on similarity in model size. For each benchmark, the figure compares the rank correlation structure of a cascade of Llama models to that of a mixed cascade consisting of models from the Llama, GPT, and Qwen families, suggesting that a cascade drawn from models of the same architectural family is more nearly Markovian.

Kendall's τ rank correlation is higher for models of similar sizes. In addition, models sharing the same architectural family (Llama, GPT, or Qwen) are more highly correlated than models of different families.

Our findings suggest that a cascade composed only of Llama models (1B-405B) satisfies the Markov assumption more exactly. Consider Figure 2.1a as an example. For the Llama cascade, Kendall's τ is highest near the heatmap's diagonal, suggesting a Markov property. By contrast, the mixed cascade composed of Llama, GPT, and Qwen models shows a more haphazard pattern. For example, the rank correlation between GPT-4o Mini and GPT-4o ($\tau = 0.55$) is higher than that between GPT-4o and Llama3 405B ($\tau = 0.54$), even though the latter pair of models

are more similar in size. Similarly, Llama3 405B is more strongly correlated with Llama3 70B ($\tau = 0.58$) than with Qwen2.5 72B ($\tau = 0.46$), even though the latter models are of near-identical size. These examples highlight that, in order for the Markov property to hold based on model size, it seems important that models share the same architectural family.

In Appendix 2F, we further verify the rank correlation patterns between different LLMs by recomputing the rank correlations only on those queries where both models answer correctly or incorrectly.

To probe the Markov property for the Llama cascade in a different way, we train logistic regressions for predicting correctness of the 8B, 70B, and 405B models based on the calibrated confidences of two ancestor models in the cascade. Specifically, we consider the immediate predecessor model (the Markov predictor) paired with each available earlier ancestor. If the Markov property holds, the Markov predictor should hold much greater significance than any other ancestor. Table 2.3 lists the results, revealing a diagonal pattern for each benchmark that confirms that the Markov predictor is usually much more significant. However, the earlier ancestor often shares statistical significance. To evaluate the significance of this finding, we also computed the magnitude of the regression coefficients corresponding to Table 2.3. The coefficients follow a similar pattern, revealing that even if multiple predictors are significant, the Markov predictor usually carries the greatest weight.

In sum, our findings suggest that for cascades composed of models sharing the same architectural family, a Markov property holds approximately, though not exactly.

Testing the Gumbel Copulas for Modeling LLM Correlations

To evaluate the goodness-of-fit of our Gumbel copula models, we first visualize the correlation between the calibrated confidences of pairs of LLMs. Figures 2.2 and 2.3 show scatterplots for several pairs of Qwen, OpenAI, and Llama models. Each scatterplot shows the copula-transformed variables

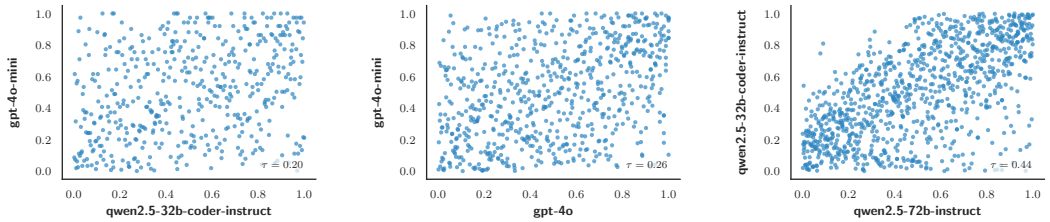
$$u = \hat{F}_n(\phi), \tag{2.19}$$

where ϕ is the calibrated confidence and $\hat{F}_n$ its empirical distribution on the test set. The marginal distribution of each u is uniform, since we restrict our copula models to the region $(\phi_{\min}, (\phi_{\max})$ of calibrated confidence where the marginal confidence distribution is smooth. Note that Figure 2.3 highlights the Markov property by showing the increasing rank correlation between Llama models of similar sizes.

Table 2.3: Verifies the Markov property for the Llama cascade by showing the results of using logistic regression to predict each model's correctness based on the calibrated confidences of two ancestor models in the cascade: the immediate predecessor model (Markov predictor) and each available earlier ancestor. For the Markov predictors, the table displays the average p values across all these logistic regressions; for the earlier ancestors, the p value corresponds to a single logistic regression. Underlined values indicate statistical significance (5% level); the lowest p values in each row are bolded. The diagonal pattern in the table suggests the Markov property.

Benchmark	**Predicted**	**$\log_{10}$ p Value of Markov Predictor vs Earlier Ancestor**			
		1B	3B	8B	70B
MMLU	8B	-2.66	**-26.86**	–	–
	70B	-0.52	-3.48	**-13.71**	–
	405B	-0.78	-2.41	-6.32	**-25.78**
MedMCQA	8B	-1.85	**-26.40**	–	–
	70B	-0.26	-2.72	**-4.35**	–
	405B	-0.23	-0.82	-2.45	**-24.63**
TriviaQA	8B	-0.14	**-22.38**	–	–
	70B	-0.58	-1.02	**-6.42**	–
	405B	-0.26	-1.88	-3.72	**-11.45**
XSum	8B	-0.72	**-1.58**	–	–
	70B	-0.97	-0.61	**-6.94**	–
	405B	-0.56	-0.50	**-2.81**	-1.62
GSM8K	8B	-2.85	**-7.48**	–	–
	70B	-0.51	-0.17	**-6.49**	–
	405B	-0.36	-0.13	**-3.22**	-2.76
TruthfulQA	8B	**-1.77**	-0.42	–	–
	70B	-0.30	-0.44	**-0.52**	–
	405B	-0.20	-0.67	-0.59	**-1.55**

We formally test the goodness-of-fit between the fitted Gumbel copulas and the test data by carrying out a Cramér-von Mises test using parametric bootstrapping, following the "Kendall's transform" approach described in Genest, Rémillard, and Beaudoin (2009). The test involves computing the univariate distribution of copula values $C_{ij}(F_i(x), F_j(x))$ for $x \sim p(x)$, using both the empirical copula and the fitted Gumbel copula. We evaluate the difference between these two distributions using the Cramér-von Mises ($\sqrt{n}$CvM) statistic and obtain a p value by parametric bootstrapping with $B = 1000$ samples. In each case, we fit the Gumbel copula on the training data ($n \approx 300$) and evaluate the p value relative to the test data ($n \approx 1000$).

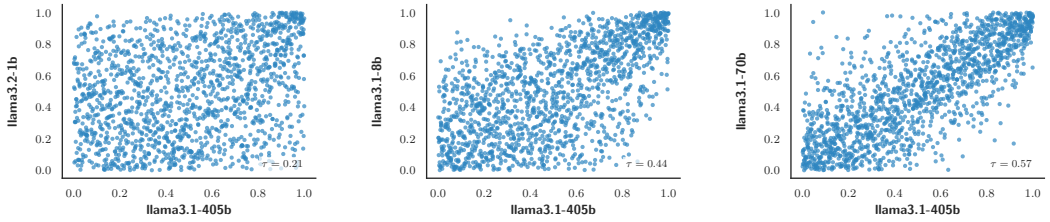


(a) GPT-4o Mini and Qwen2.5 32B Coder on GSM8K

(b) GPT-4o Mini and GPT-4o on MedMCQA

(c) Qwen2.5 32B Coder and Qwen2.5 72B on TriviaQA

Figure 2.2: Correlations between the calibrated confidences of selected pairs of LLMs on different benchmarks, showing a range of rank correlations between models. The Kendall's τ rank correlation, shown in the bottom right corner, ranges from $\tau = 0.20$ to $\tau = 0.44$.



(a) 1B-405B

(b) 8B-405B

(c) 70B-405B

Figure 2.3: Correlations between the calibrated confidences of smaller Llama models (1B, 8B, 70B) (y axis) and the 405B model (x axis) on MMLU. The increasing rank correlation suggests a Markov property based on model size. The Kendall's τ rank correlation, shown in the bottom right corner, increases from $\tau = 0.21$ to $\tau = 0.57$.

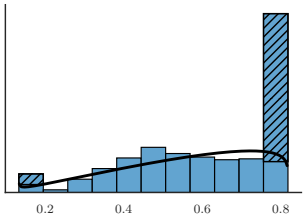
Table 2.4 breaks down the results by benchmark for two groups of models (Llama models vs OpenAI & Qwen models). Each reported number is based on considering all pairs of models within each group, regardless of similarities in size. There are 10 pairs of Llama models and 6 pairs of Qwen and OpenAI models. The results show that for the Llama models, the fitted Gumbel copulas closely match the empirical correlation structures between pairs of models on the test set, since the overall rejection rate of the null hypothesis is only 1.7%, well below the 5% rejection rate expected by chance. In addition, the $\sqrt{n}$CvM statistic is only 0.003 on average.

For the group of Qwen and OpenAI models, we observe higher rejection rates. The overall rejection rate of 22.2% suggests that the Gumbel copula model does not fit the data exactly. However, the average $\sqrt{n}$CvM value of 0.008 suggests that the fit is adequate.

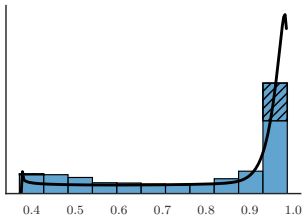
Table 2.4: Shows the goodness-of-fit of our Gumbel copula models for modeling pairwise correlations between LLMs, based on a Cramér-von Mises ($\sqrt{n}$CvM) test using parametric bootstrapping. We report the $\sqrt{n}$CvM value, the number of null hypothesis rejections (out of 10 and 6 model pairs for the Llama and Qwen & OpenAI groups, respectively), the percentage of rejections, as well as the geometric and arithmetic mean of p values.

	Llama Models					**Qwen & OpenAI Models**				
Benchmark	$\sqrt{n}$CvM	# Rej.	% Rej.	$(\prod p)^{\frac{1}{n}}$	$\frac{1}{n}\sum p$	$\sqrt{n}$CvM	# Rej.	% Rej.	$(\prod p)^{\frac{1}{n}}$	$\frac{1}{n}\sum p$
MMLU	0.002	0	0.0	0.569	0.591	0.011	4	66.7	0.058	0.121
MedMCQA	0.004	1	10.0	0.394	0.560	0.004	0	0.0	0.397	0.444
TriviaQA	0.002	0	0.0	0.638	0.709	0.012	2	33.3	0.078	0.187
XSum	0.004	0	0.0	0.405	0.480	0.002	0	0.0	0.704	0.733
GSM8K	0.002	0	0.0	0.688	0.757	0.016	2	33.3	0.032	0.157
TruthfulQA	0.001	0	0.0	0.961	0.963	0.002	0	0.0	0.800	0.812
Average	0.003	0	1.7	0.609	0.677	0.008	1	22.2	0.345	0.409

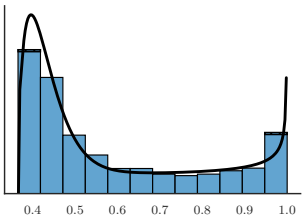
Testing the Discrete-Continuous Marginal Confidence Distributions



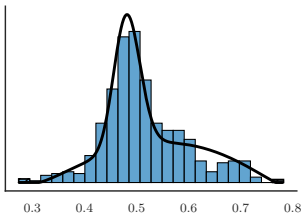
(a) GPT-4o Mini on TruthfulQA



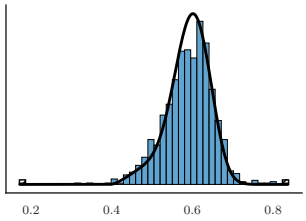
(b) GPT-4o on MedMCQA



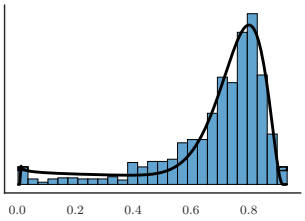
(c) Llama3.2 3B on MMLU



(d) Llama3.1 8B on TruthfulQA



(e) Llama3.2 1B on GSM8K



(f) Qwen2.5 32B Coder on XSum

Figure 2.4: Selection of trained marginal distributions (fitted on $n \approx 300$ training data), with histograms of the test data ($n \approx 1000$). Histogram areas shaded with hatch patterns (especially in (a) and (b) indicate the contributions of discrete probability masses in our models.

First, we visualize the agreement between the fitted continuous-discrete mixtures of scaled beta distributions and the histograms of calibrated confidence values on the test set. To construct these plots, we first train the calibrators and marginal

distributions on the training set ($n \approx 300$ examples).[4] We then compute the calibrated confidence on the test set ($n \approx 1000$) using the trained calibrators. Figure 2.4 suggests that the fitted marginals align well with the calibrated confidence values on the test data.

Each histogram displays the discrete masses $\phi_{\min}$ and $\phi_{\max}$ of the fitted marginal distributions by shading corresponding areas on the first and last bars of each histogram. We observe in Figure 2.4a that the discrete probability masses are especially pronounced for GPT-4o Mini on TruthfulQA and GPT-4o on MedMCQA. The trend that the OpenAI GPT models often report certainty also holds for other benchmarks, as Table 2.1 shows.

Table 2.5: Shows the goodness-of-fit of our discrete-continuous mixtures of scaled beta distributions for modeling the marginal distributions of calibrated LLM confidence. We computed p values for the square-rooted Cramér-von Mises ($\sqrt{\text{CvM}}$) statistic using parametric bootstrapping with $B = 1000$ samples. The $\sqrt{\text{CvM}}$ statistic and its p value were computed on the test set ($n \approx 1000$), whereas the marginal distributions were fitted on the training set ($n \approx 300$). We highlight $p < 0.05$ with an underline and $p < 0.001$ with **bold** font. Additionally, we **bold** the largest $\sqrt{\text{CvM}}$ value within each column. Highlighted values indicate the greatest discrepancies with our model.

	MMLU		**MedMCQA**		**TriviaQA**		**XSum**		**GSM8K**		**TruthfulQA**	
Model	$\sqrt{\text{CvM}}$	p	$\sqrt{\text{CvM}}$	p	$\sqrt{\text{CvM}}$	p	$\sqrt{\text{CvM}}$	p	$\sqrt{\text{CvM}}$	p	$\sqrt{\text{CvM}}$	p
llama3.2-1b	0.031	**0.000**	0.025	0.015	0.018	0.117	0.036	0.001	0.026	0.015	0.025	0.109
llama3.2-3b	0.014	0.144	**0.115**	**0.000**	0.043	**0.000**	0.020	0.076	0.020	0.071	0.030	0.053
llama3.1-8b	0.016	0.066	0.088	**0.000**	0.022	0.033	0.037	**0.000**	0.016	0.163	0.022	0.181
llama3.1-70b	**0.048**	**0.000**	**0.137**	**0.000**	**0.057**	**0.000**	**0.070**	**0.000**	0.038	**0.000**	0.044	0.002
llama3.1-405b	0.024	0.004	0.113	**0.000**	0.028	0.008	0.027	0.009	0.034	0.001	0.036	0.019
gpt-4o-mini	0.032	**0.000**	0.060	**0.000**	0.008	0.441	0.020	0.077	0.026	0.016	0.028	0.072
qwen2.5-32b-c	0.036	**0.000**	0.069	**0.000**	0.040	**0.000**	0.020	0.067	0.028	0.010	0.023	0.160
qwen2.5-72b	0.028	0.001	0.073	**0.000**	0.040	**0.000**	0.041	**0.000**	0.004	0.678	0.036	0.018
gpt-4o	0.029	**0.000**	0.100	**0.000**	0.046	**0.000**	0.026	0.013	0.036	0.001	**0.065**	**0.000**
Average	0.029	0.024	0.087	0.003	0.034	0.066	0.033	0.027	0.025	0.106	0.034	0.068

We formally test the goodness-of-fit of the marginal distributions by computing the square-rooted Cramér-von Mises statistic

$$\sqrt{\text{CvM}} = \sqrt{\int (\hat{F}_n^{\text{test}}(x) - F(x|\hat{\boldsymbol{\theta}}))^2 \, \mathrm{d}F(x|\hat{\boldsymbol{\theta}})}, \tag{2.20}$$

where $\hat{F}_n^{\text{test}} = \frac{1}{n}\sum_{i=1}^{n} \delta_{\Phi(x_i)}$ is the empirical distribution of the calibrated confidence on the test data, and $F(\cdot|\boldsymbol{\theta})$ is our marginal distribution model (2.7) with $\boldsymbol{\theta} =$

[4] We do not consider it necessary to train the calibrators and the marginal confidence distributions on separate training data sets, since the calibrators model $p(y|x)$ and the marginal distributions model $p(x)$.

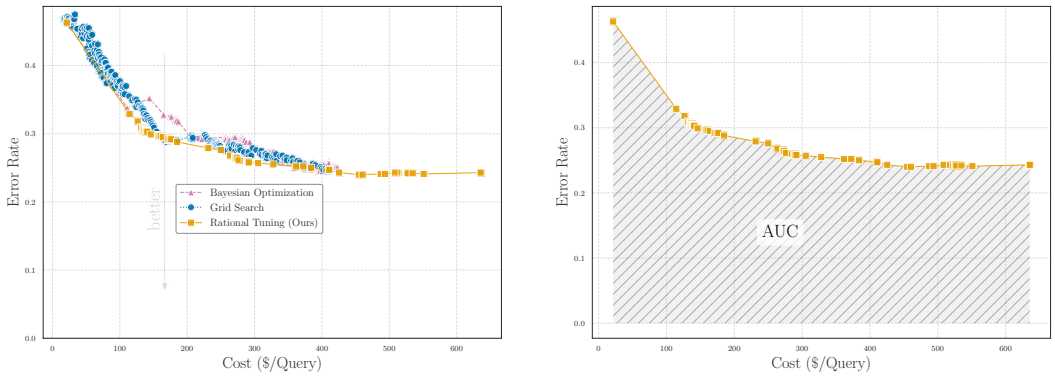
Table 2.6: Shows the goodness-of-fit of our discrete-continuous mixtures of scaled beta distributions for modeling the marginal distributions of calibrated LLM confidence, after re-fitting the marginal distributions on the test set. We computed p values for the square-rooted Cramér-von Mises ($\sqrt{\text{CvM}_\text{r}}$) statistic using parametric bootstrapping with $B = 1000$ samples. We highlight $p < 0.05$ with an underline and $p < 0.001$ with **bold** font. Additionally, we **bold** the largest $\sqrt{\text{CvM}_\text{r}}$ value within each column. Highlighted values indicate the greatest discrepancies with our model.

	MMLU		MedMCQA		TriviaQA		XSum		GSM8K		TruthfulQA	
Model	$\sqrt{\text{CvM}_\text{r}}$	p	$\sqrt{\text{CvM}_\text{r}}$	p	$\sqrt{\text{CvM}_\text{r}}$	p	$\sqrt{\text{CvM}_\text{r}}$	p	$\sqrt{\text{CvM}_\text{r}}$	p	$\sqrt{\text{CvM}_\text{r}}$	p
llama3.2-1b	0.018	0.046	0.020	0.085	0.005	0.986	0.006	0.935	0.018	0.126	0.008	0.970
llama3.2-3b	0.009	0.461	0.010	0.608	0.036	**0.000**	0.009	0.666	0.010	0.566	0.013	0.661
llama3.1-8b	0.012	0.248	0.010	0.574	0.011	0.492	0.009	0.688	0.006	0.947	0.010	0.846
llama3.1-70b	0.016	0.072	0.018	0.121	0.024	0.029	0.017	0.141	0.023	0.037	0.015	0.460
llama3.1-405b	0.015	0.133	0.011	0.498	0.017	0.130	0.006	0.933	0.031	0.002	0.019	0.290
gpt-4o-mini	0.004	0.928	0.007	0.853	0.003	0.913	0.011	0.393	0.009	0.549	0.014	0.548
qwen2.5-32b-c	0.011	0.282	0.012	0.355	0.020	0.070	0.022	0.047	0.015	0.217	0.013	0.600
qwen2.5-72b	0.018	0.039	0.016	0.157	0.028	0.005	0.014	0.301	0.002	0.966	0.008	0.970
gpt-4o	0.011	0.273	0.024	0.315	0.041	**0.000**	0.013	0.367	0.030	0.005	0.021	0.759
Average	0.013	0.276	0.014	0.396	0.021	0.292	0.012	0.497	0.016	0.379	0.013	0.678

$(\phi_{\min}, \phi_{\max}, w_{\min}, w_{\max}, \pi, \alpha_1, \beta_1, \alpha_2, \beta_2)$. In Tables 2.5 and 2.6, we report (2.20) both for $\hat{\boldsymbol{\theta}}$ estimated from the training data ($\sqrt{\text{CvM}}$), and for $\hat{\boldsymbol{\theta}}$ re-fitted on the test data ($\sqrt{\text{CvM}_\text{r}}$). The reason we report $\sqrt{\text{CvM}_\text{r}}$ is to evaluate whether deficiencies in the fit arise from a bias problem, rather than a variance problem. To compute p values for (2.20), we use parametric bootstrapping with $B = 1000$ samples.

Table 2.5 indicates a close fit between the trained marginal distributions and the empirical distributions of the calibrated confidences on the test data, with an average $\sqrt{\text{CvM}}$ value of 4%. However, 74% of tests reject the null hypothesis at the $p < 0.05$ level, suggesting that our model does not exactly match the data. When refitting the marginals on the test data, the average $\sqrt{\text{CvM}}$ value falls to 1.5% and a much lower 18.5% of tests reject the null hypothesis. Even on the refitted data, this overall rejection rate of 18.5% is significantly higher than the 5% we would expect by chance. We conclude that our marginal distribution model fits the empirical data well, as judged by a low $\sqrt{\text{CvM}}$ value, but it clearly does not capture the true distribution of calibrated confidences exactly.

Notably, the results for the refitted marginals show that the quality of the fit strongly depends on the benchmark. Specifically, TriviaQA displays a much poorer fit than the other benchmarks. For many of the LLMs, TriviaQA's low difficulty (as judged by a 90%+ test accuracy for many models) explains the poor fit. The presence of a sharp peak of calibrated confidences near $\phi_{\max}$ presumably raises

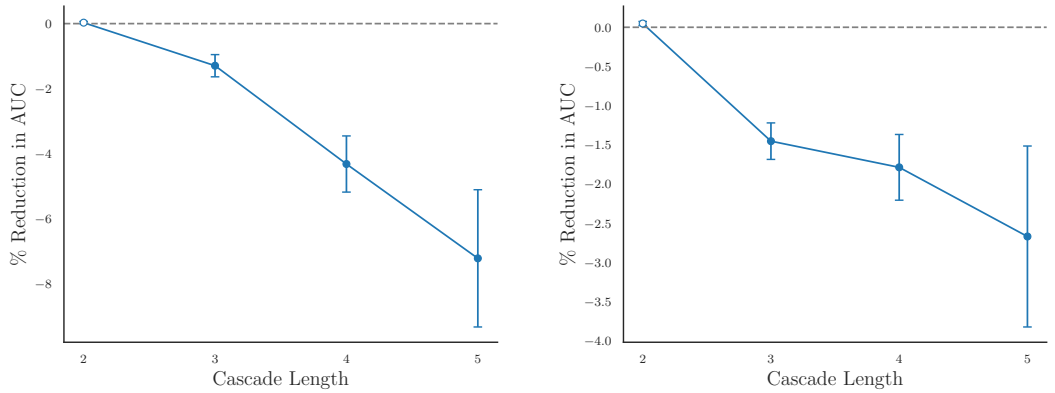


(a) Error-cost curves (b) Area under the error-cost curve

Figure 2.5: Performance evaluation via the area under the error-cost curve (AUC). (a) Error-cost curves computed on the MedMCQA test set for the Llama3 3B → 8B → 70B → 405B cascade. (b) Illustration of the area under the error-cost curve (AUC).

the number of training samples required to precisely estimate the shape of the distribution. In addition, the ability of the beta distribution to fit sharply peaked unimodal distributions may be inherently limited. We hypothesize that these factors may explain the high p values despite rather low $\sqrt{\text{CvM}}$ values.

Rational Tuning of Confidence Thresholds



(a) Relative to Bayesian optimization (b) Relative to grid search

Figure 2.6: Reduction in the area under the error-cost curve (AUC) on the test set when using our Rational Tuning framework to select confidence thresholds, as a function of cascade length. In (a), we compare against a Bayesian optimization baseline, while in (b) we compare against high-resolution grid search. For longer cascades, our method outperforms both baselines by larger margins. Error bars show $\pm 1\sigma$ of the mean percentage change, and filled markers indicate statistical significance.

Benchmark	AUC ↓			RT (Ours) vs GS		RT (Ours) vs BO	
	RT (Ours)	GS	BO	$\%\Delta_{GS}\downarrow$	p_{GS}	$\%\Delta_{BO}\downarrow$	p_{BO}
MMLU	0.288	**0.288**	0.294	0.02	2.8×10^{-1}	-2.30	$\mathbf{4.3 \times 10^{-3}}$
MedMCQA	**0.381**	0.384	0.389	-0.79	$\mathbf{5.5 \times 10^{-3}}$	-2.14	$\mathbf{3.0 \times 10^{-2}}$
TriviaQA	**0.181**	0.182	0.183	-0.74	$\mathbf{6.0 \times 10^{-3}}$	-1.58	$\mathbf{1.0 \times 10^{-2}}$
XSum	**0.409**	0.414	0.416	-1.48	$\mathbf{2.0 \times 10^{-6}}$	-1.92	$\mathbf{8.9 \times 10^{-4}}$
GSM8K	**0.158**	0.162	0.160	-2.68	$\mathbf{1.0 \times 10^{-5}}$	-1.15	$\mathbf{1.6 \times 10^{-2}}$
TruthfulQA	**0.436**	0.437	0.438	-0.26	$\mathbf{3.8 \times 10^{-2}}$	-0.46	7.9×10^{-2}
Average	**0.309**	0.311	0.313	-0.99	–	-1.60	–

Table 2.7: Area under the error-cost curve (AUC) on the test set, showing that our Rational Tuning ("RT") framework for selecting confidence thresholds consistently outperforms both a Bayesian optimization baseline ("BO") and high-resolution grid search ("GS"). The mean percentage changes ($\%\Delta$) are statistically significant at the $p < 0.05$ on almost all benchmarks, as measured by Wilcoxon rank-sum tests paired by cascade (highlighted in bold).

In this section, we examine the performance and runtime scaling of our continuous optimization-based algorithm (2.11) for selecting optimal confidence thresholds. We consider all 26 possible cascades of length $k \geq 2$ composed of Meta's Llama models (1B, 3B, 8B, 70B, and 405B). We evaluate against Bayesian optimization and high-resolution grid search baselines on six benchmarks (MMLU, MedMCQA, XSum, TriviaQA, GSM8K, TruthfulQA) spanning general-purpose knowledge and reasoning, domain-specific QA, text summarization, open-ended QA, mathematical reasoning, and the ability to avoid hallucinations on adversarial questions.

Performance metrics: we evaluate the area under the error-cost curve (AUC) on the test set. Specifically, computing the AUC means plotting the test error (y axis) against the expected inference cost in dollars/query (x axis) and evaluating the integral of this curve. Figure 2.5a shows an example error-cost curve and Figure 2.5b highlights the computation of AUC. We normalize the cost values to lie between 0 and 1, resulting in AUC scores between 0 and 1 (error rate $\times$ normalized cost). Broadly, a 1% reduction in AUC means that the error rate is 1% lower at the same inference cost (on average).

In addition, we measure how the runtime for finding optimal confidence thresholds scales with the length of the cascade and the desired resolution of the error-cost curve on the x axis, *i.e.*, how densely we sample the optimal thresholds. We have not overly optimized our code and mainly aim to contrast asymptotic scaling behavior.

Bayesian optimization baseline: this baseline runs Bayesian optimization with a Gaussian process surrogate function, via the HEBO package (Cowen-Rivers et al., 2022, Shahriari et al., 2016). The Bayesian optimization minimizes (2.11), in an analogous manner to our Markov-copula ("Rational Tuning") approach. We run HEBO for as many iterations as needed until the change in loss between successive iterations is below a numerical tolerance ($\epsilon = 10^{-5}$). In practice, we found that the final change in loss is typically 0.0. Following the practical guidance of HEBO's authors[5], we use four parallel suggestions during each iteration. We adaptively interpolate the optimal thresholds computed by HEBO in the same way we do for Rational Tuning (see Equation (2.13)).

High-resolution grid search baseline: this baseline selects optimal confidence thresholds by searching over adaptive grids computed from the model-specific quantiles of calibrated confidence. Specifically, in each dimension the grid ranges from $\phi_{\min}$ to $\phi_{\max}$ in increments of 2.5% probability mass. This results in considering 40^{k-1} candidate threshold combinations for cascades with k models, ranging from 40 candidates for a two-model cascade to 40^4 = 2,560,000 candidates for a five-model cascade. After scoring all candidate threshold combinations, we use the Skyline operator (implemented in the Python package `paretoset`[6]) to filter the candidate threshold vectors down to the Pareto-optimal set (Börzsönyi, Kossmann, and Stocker, 2001). A candidate threshold vector $\boldsymbol{\theta} = (\phi_1, ..., \phi_{k-1})$ is *Pareto-optimal* if its performance metrics $(\mathbb{P}_{\boldsymbol{\theta}}(\text{Correct}), \mathbb{E}_{\boldsymbol{\theta}}[\text{Cost}])$ are not dominated by any other candidate threshold vector $\boldsymbol{\theta'}$ in the sense that $\mathbb{P}_{\boldsymbol{\theta'}}(\text{Correct}) > \mathbb{P}_{\boldsymbol{\theta}}(\text{Correct})$ and $\mathbb{E}_{\boldsymbol{\theta'}}[\text{Cost}] < \mathbb{E}_{\boldsymbol{\theta}}[\text{Cost}]$.

Figure 2.6 shows that our Rational Tuning framework for selecting confidence thresholds results in lower AUC on the test set compared to the baselines. Each point on the plot shows the average percentage reduction in AUC for all cascades of a given length k, averaged across all benchmarks. As cascade length k grows, our method outperforms the baselines by a larger margin. For example, the mean reduction in AUC compared to Bayesian optimization is 4.3% for $k \geq 3$; 5.8% for $k \geq 3$; and 7.2% for $k = 5$. The corresponding performance gains relative to high-resolution grid search are 2.0%, 2.2%, and 2.7%. We computed statistical

[5] `github.com/huawei-noah/HEBO/tree/master/HEBO`. Accessed April 6, 2025.

[6] Open-source implementation available at `github.com/tommyod/paretoset`. Accessed January 13, 2025.

significance of these percentage differences using a Wilcoxon rank-sum test paired by cascade.

We hypothesize that the performance gains of Rational Tuning relative to Bayesian optimization stem from the fact that our framework applies a (mostly correct) inductive assumption about the correlation structure of LLM cascades, whereas Bayesian optimization is a general-purpose algorithm for black-box optimization. Section 2.4 corroborates this hypothesis by presenting more significant performance gains in the low-sample limit with $n \leq 30$ training examples.

By contrast, it is not surprising that grid search performs worse as k increases, since searching for effective threshold combinations by trial and error suffers from the curse of dimensionality.

Table 2.7 presents the results broken down by benchmark rather than cascade length. The table shows that Rational Tuning consistently outperforms Bayesian optimization and grid search across benchmarks, independent of cascade length. The only benchmark mark where we report a tie is MMLU. On almost all benchmarks, the reductions in AUC are statistically significant at the $p < 0.05$ level.

Performance in the Low-Sample Limit

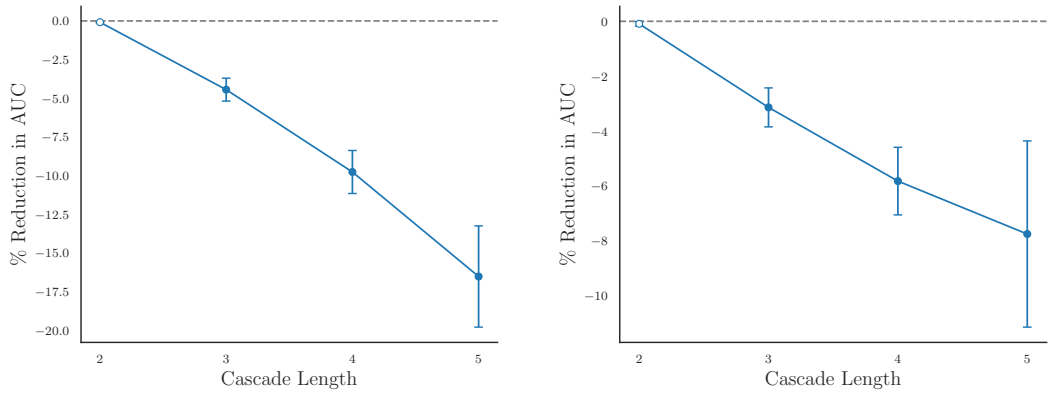
Our Rational Tuning methodology relies on a small labeled training data set consisting of LLM confidence scores and corresponding binary correctness labels. Since labeled data is scarce in many applications, we supplement our main experiment ($n \approx 300$ training examples) with a study of the low-sample limit. Here, we re-run our experiment for $n \leq 30$ training examples. For each benchmark, we ensure a balanced subsample with both correct and incorrect answers from each model by sampling training examples in pairs (one correct, one incorrect for each model) for a fixed number of iterations, with a target of $n = 30$ examples. Since collisions may occur, the final number of sampled training examples lies between 20 and 30, depending on the benchmark.

Figure 2.7 displays the results, revealing that Rational Tuning significantly outperforms the Bayesian optimization and grid search baselines as cascade length grows. On cascades with $k \geq 3$ models, the average performance gain is 10.2% relative to Bayesian optimization, and 5.6% relative to high-resolution grid search.

Table 2.8 breaks down these results by benchmark. We see that Rational Tuning outperforms the baselines on each benchmark except for TruthfulQA, where

high-resolution grid search performs best. On the other benchmarks (MMLU, MedMCQA, TriviaQA, XSum, and GSM8K), the performance gains of our method are statistically significant at the $p < 0.05$ level, according to Wilcoxon rank-sum tests paired by cascade.

Our interpretation of these results is that our Rational Tuning framework benefits from making inductive assumptions about the interactions between the error rates of different LLMs. Crucially, fitting these inductive assumptions to empirical data requires only few observations: since each copula model depends on a single scalar correlation parameter $\theta \in \mathbb{R}$, our method requires only $k - 1$ parameters to model the interactions between the error rates of k different LLMs.



(a) Relative to Bayesian optimization

(b) Relative to grid search

Figure 2.7: Reduction in the area under the error-cost curve (AUC) as cascade length grows, in the low-sample limit ($n \leq 30$ training examples), when using our Rational Tuning framework. In (a), we compare against a Bayesian optimization baseline, while in (b) we compare against high-resolution grid search. Our method increasingly outperforms the baselines as cascade length grows. Error bars show $\pm 1\sigma$ of the mean percentage change, and filled markers indicate statistical significance.

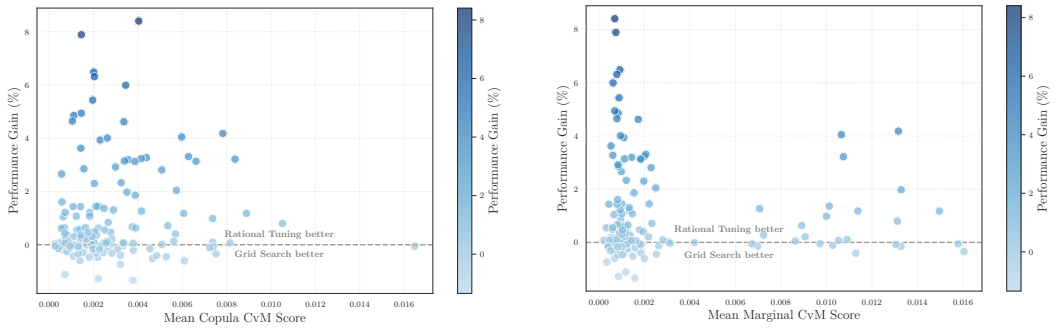
Sensitivity to Statistical Assumptions

Our implementation of Rational Tuning models uses mixtures of beta distributions to model the marginal distribution of confidence scores, and Gumbel copulas to model pairwise correlations between LLMs. In Section 2.4, we quantify the deviation between these modeling assumptions and the true empirical distributions via Cramér-von Mises (CvM) statistics.

Figure 2.8 shows the sensitivity of Rational Tuning's performance gains (relative to the high-resolution grid search baseline) to the mean CvM statistics for each

Benchmark	AUC ↓			RT (Ours) vs GS		RT (Ours) vs BO	
	RT (Ours)	GS	BO	$\%\Delta_{GS}$ ↓	p_{GS}	$\%\Delta_{BO}$ ↓	p_{BO}
MMLU	**0.293**	0.308	0.316	-5.51	$\mathbf{8.0 \times 10^{-6}}$	-7.74	$\mathbf{1.0 \times 10^{-6}}$
MedMCQA	**0.399**	0.416	0.419	-4.18	$\mathbf{1.1 \times 10^{-5}}$	-4.72	$\mathbf{5.0 \times 10^{-6}}$
TriviaQA	**0.197**	0.200	0.203	-2.11	$\mathbf{1.5 \times 10^{-2}}$	-3.26	$\mathbf{5.1 \times 10^{-3}}$
XSum	**0.422**	0.431	0.442	-2.42	$\mathbf{2.6 \times 10^{-2}}$	-5.06	$\mathbf{4.2 \times 10^{-4}}$
GSM8K	**0.187**	0.195	0.197	-3.41	$\mathbf{2.5 \times 10^{-2}}$	-4.21	$\mathbf{2.1 \times 10^{-4}}$
TruthfulQA	0.449	**0.442**	0.452	1.60	1.0×10^{0}	-0.73	1.2×10^{-1}
Average	**0.325**	0.332	0.338	-2.67	–	-4.29	–

Table 2.8: Area under the error-cost curve (AUC) in the low-sample limit ($n \leq 30$ training examples), showing that our Rational Tuning ("RT") framework for selecting confidence thresholds consistently outperforms both a Bayesian optimization baseline ("BO") and high-resolution grid search ("GS"). The mean percentage changes ($\%\Delta$) are statistically significant at the $p < 0.05$ on almost all benchmarks, as measured by Wilcoxon rank-sum tests paired by cascade (highlighted in bold).



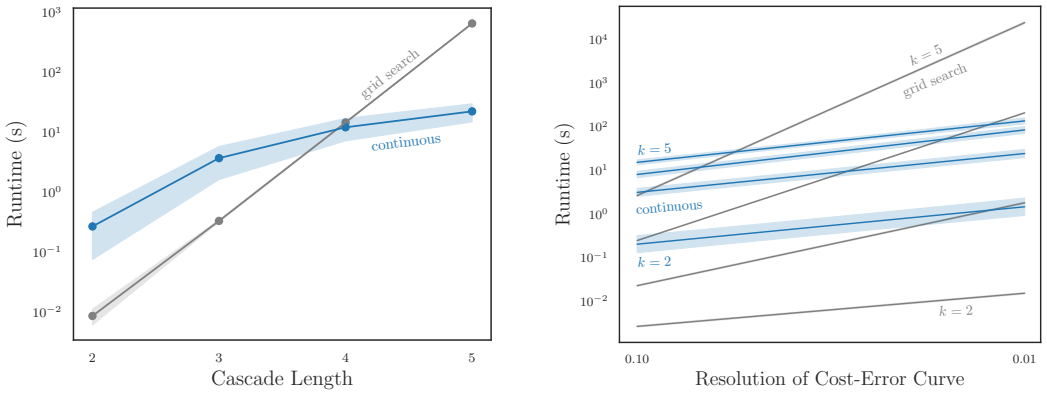
(a) Copula $\sqrt{n}$CvM (b) Marginal $\sqrt{\text{CvM}}$

Figure 2.8: Sensitivity of Rational Tuning's performance gains to the Cramér-von Mises (CvM) test statistics (lower is better). Overall, performance appears to be more sensitive to mis-specification of the copula model.

cascade. For example, if $M_1 \to ... \to M_k$ has marginal $\sqrt{\text{CvM}}$ scores $\sigma_1, ..., \sigma_k$ and copula $\sqrt{n}$CvM scores $\sigma_{1,2}, \sigma_{2,3}, ..., \sigma_{k-1,k}$, then the mean marginal and copula CvM scores are $\overline{\sigma}_{\text{marginal}} = \frac{1}{k}\sum_{i=1}^{k}\sigma_i$, and $\overline{\sigma}_{\text{copula}} = \frac{1}{k}\sum_{i=2}^{k}\sigma_{i-1,i}$, respectively.

Figures 2.8a and 2.8b suggest that lower CvM divergences improve the relative performance of Rational Tuning. This effect is more pronounced for the copula statistics rather than the marginal statistics, highlighting the importance of correctly modeling the correlations between LLMs. In both plots, the light blue data points with little performance gain despite excellent CvM values are heavily enriched for two-model cascades. For cascades with $k = 2$ models, Rational Tuning generally performs on par with grid search or Bayesian optimization.

The plots also suggest some robustness to deviations from the statistical assumptions.



(a) The runtime of grid search grows exponentially in the cascade length, whereas the runtime of our method grows as a low-order polynomial (semilog-y plot).

(b) The runtime of our method always grows linearly in the desired resolution h at which the error-cost curve is sampled along the cost axis, independent of the cascade length k. However, the runtime of grid search scales as h^{k-1} in h and k (log-log plot).

Figure 2.9: Shows runtime scaling for computing the full error-cost curve, comparing our continuous-optimization based algorithm ("continuous," blue) to grid search ("grid search," gray). Our method scales much more favorably as the cascade length grows, and as the error-cost curve is sampled more densely along the cost axis. The shading shows $\pm 1\sigma$ of the observed data points.

We are eager to explore Rational Tuning's robustness to model mis-specification in greater detail in subsequent work.

Computational Scaling

Moving on to a comparison of computational complexity, Figure 2.9 shows that the runtime of our Rational Tuning framework for finding optimal confidence thresholds scales much more favorably compared to grid search, both in the length of the cascade k as well as the desired resolution h of the error-cost curve. Here, the resolution h refers to the density at which we sample the optimal error-cost curve along the cost-axis. For grid search, h is simply the reciprocal of the number of grid points in each dimension. For our method, h is the reciprocal of the number of times we solve the optimization problem (2.11). In other words, $h = 1/|\Lambda|$, where Λ is the set of cost sensitivities we consider in (2.11).

We omit Bayesian optimization from Figure 2.9, since we observed longer runtimes (10-1000x longer) that exhibit less clear scaling with the length k of the cascade. Specifically, the average number of iterations until convergence increases from 2.9

for $k = 2$ to 5.3 for $k = 4$, then drops back to 4.0 for $k = 5$. Across the data, the minimum and maximum number of iterations required until convergence are 2 and 14.

Practical Guidelines

Our experiments suggest that Rational Tuning is the preferred methodology for tuning confidence thresholds for longer cascades ($k \geq 3$) with multiple deferral thresholds. However, for two-model cascades ($k = 2$) with a single deferral threshold, the Markov assumption is void (as there are only two models). In this setting, the nonparametric nature of one-dimensional grid search should give the most reliable results.

When applying Rational Tuning for cascades with $k \geq 3$ models, we recommend visually inspecting the match between the assumed probabilistic model and the empirical data. Specifically, we recommend the following visual diagnostics:

- **Marginal distributions**: compare histograms of the fitted and empirical distributions (as in Figure 2.4) and verify that the overall fit is adequate.
- **Pairwise correlations**: construct copula plots as in Figures 2.2 and 2.3. Compare these plots to random samples from a Gumbel copula with correlation parameter $\hat{\theta} = \frac{1}{1-\hat{\tau}}$, where $\hat{\tau}$ is the empirical rank correlation.
- **Markov assumption**: construct rank correlation plots as in Figure 2.1. Verify that rank correlations are strong near the diagonal.

In addition, it is important to assess the expected calibration error (ECE) of the confidence scores. We recommend computing the ECE using quantile binning with 10 or 20 bins; ideally, the ECE should not exceed 10%.

2.5 Conclusion

We have presented a framework for rationally tuning the confidence thresholds of LLM cascades using continuous optimization. Our approach is based on a parametric probabilistic model for the calibrated confidences of a sequence of LLMs. This probabilistic model is based on a Markov factorization, which accounts for pairwise correlations between the error rates of different LLMs using copulas, yielding a data-efficient approach. Goodness-of-fit analyses spanning 10 LLMs and 6 benchmarks have shown good agreement with the test data.

Importantly, our probabilistic model yields analytical expressions for a cascade's error rate and expected inference cost. These expressions are differentiable with respect to the cascade's confidence thresholds, making continuous optimization possible. Compared to selecting confidence thresholds using Bayesian optimization and high-resolution grid search, our Rational Tuning framework yields more favorable error-cost trade-offs as cascade length grows, outperforming the baselines by up to 7.2% when using $n \approx 300$ labeled training examples. In the low-sample limit ($n \leq 30$ training examples), the performance gains reach up to 16.5%, suggesting that our framework's inductive assumptions about the interactions between the error rates of different LLMs improve sample efficiency.

Building on these promising results, an interesting direction would be to apply our probabilistic modeling framework to LLM routing, in which a central routing model sends a query to the most suitable LLM in a single step, avoiding cumulative cost increases as the query propagates down a cascade. Since cumulative cost increases are especially severe for longer cascades (at which our methodology excels), the routing setting may more effectively leverage Rational Tuning's capacity for modeling dependencies between arbitrarily many distinct LLMs. For instance, suppose the routing decision depends on noisy estimates $\hat{\phi}_1, ..., \hat{\phi}_n$ of the LLMs' true calibrated confidences. In this case, balancing the noisy observations $\hat{\phi}_i$ against their probabilistic expectations $\mathbb{E}[\hat{\phi}_i|\hat{\phi}_1, ..., \hat{\phi}_{i-1}, \hat{\phi}_{i+1}, ...\hat{\phi}_n]$ may lead to more effective routing decisions.

Ultimately, our results point to a larger vision for the future of deploying LLMs. Using probabilistic models, we will be able to adaptively select the most suitable model to answer each query, improving both reliability and performance. Additionally, probabilistic modeling will enable us to anticipate the performance of a system of LLMs under different conditions, making it possible to seamlessly adapt the system as conditions shift. We are excited to further pursue this line of research in subsequent work.

Chapter 3

ECONOMIC EVALUATION

Zellinger, Michael J. and Matt Thomson (2025). "Economic Evaluation of LLMs." In: *arXiv preprint*. arXiv: 2507.03834 [cs.AI].

Abstract: *Practitioners often navigate LLM performance trade-offs by plotting Pareto frontiers of optimal accuracy-cost trade-offs. However, this approach offers no way to compare between LLMs with distinct strengths and weaknesses: for example, a cheap, error-prone model vs a pricey but accurate one. To address this gap, we propose economic evaluation of LLMs. Our framework quantifies an LLM's performance trade-off as a single number based on the economic constraints of a concrete use case, all expressed in dollars: the cost of making a mistake, the cost of incremental latency, and the cost of abstaining from a query. We apply our economic evaluation framework to compare the performance of reasoning and non-reasoning models on difficult questions from the MATH benchmark, discovering that reasoning models offer better accuracy-cost tradeoffs as soon as the economic cost of a mistake exceeds $0.01. In addition, we find that single large LLMs often outperform cascades when the cost of making a mistake is as low as $0.1. Overall, our findings suggest that when automating meaningful human tasks with AI models, practitioners should typically use the most powerful available model, rather than attempt to minimize AI deployment costs, since deployment costs are likely dwarfed by the economic impact of AI errors.*

3.1 Introduction

Large language models (LLM) are commonly evaluated based on their accuracy, cost, latency, and other metrics (Liang et al., 2023). Practitioners commonly display available models on an accuracy-cost scatter plot to identify models offering the best accuracy-cost trade-offs (Hu et al., 2024). These optimal trade-offs are referred to as a "Pareto frontier" (Jin, 2006; Branke et al., 2008). Unfortunately, Pareto frontiers do not provide a way to rank models with distinct strengths and weaknesses. For example, it is not possible to compare a cheap, error-prone model against a pricey but accurate one. However, practitioners often face such dilemmas (Hammond, 2024).

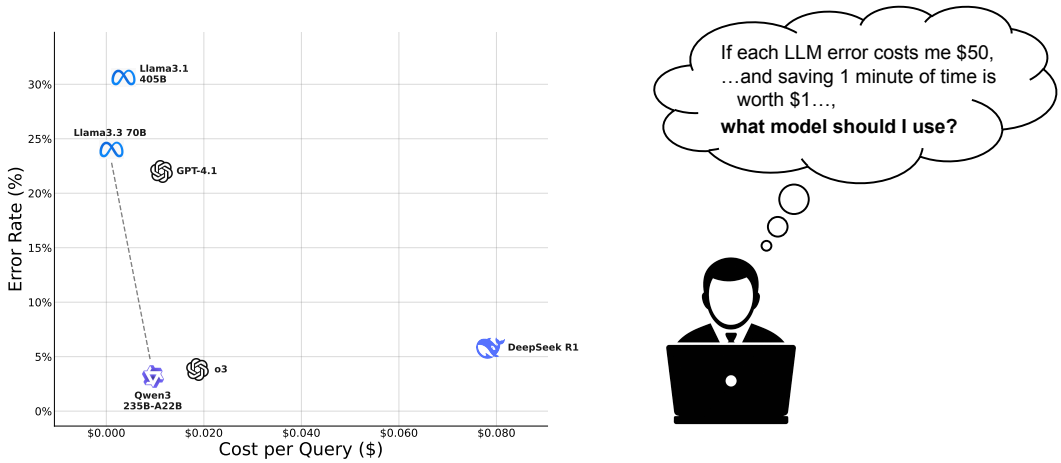


Figure 3.1: Pareto frontiers of LLM performance do not reveal which model is best-suited for a given use case—a problem often faced by practitioners.

To address this problem, we propose an economic framework for evaluating AI models, which enables practitioners to identify the single best model for their use case, even when balancing competing objectives such as accuracy, cost, and latency. Drawing on the well-understood interpretation of Lagrange multipliers as shadow prices (Bertsekas, 1999), we model a concrete use cases in terms of economic constraints expressed in dollars: the cost of making a mistake (the *price of error*), the cost of incremental latency (the *price of latency*), and the cost of abstaining from a query (the *price of abstention*).

Given these economic parameters, our framework determines the optimal LLM. As an example, suppose a hospital deploys LLMs for medical note-taking. For this use case, the *price of error* likely exceeds $100, but the *price of latency* may be low, perhaps $1 per minute of incremental latency (equivalent to human wages of $60/hour). By contrast, a natural language search engine for an e-commerce platform faces strikingly different constraints. In this domain, the price of error is much lower (perhaps $1 per error) but each additional 100 milliseconds of latency are costly, since unresponsive websites drive away consumers (Kohavi and Longbotham, 2007).

We demonstrate the practical utility of economic evaluation by addressing two important questions for practical LLM deployments. First, we ask whether reasoning or non-reasoning LLMs are optimal on complex problem-solving tasks when taking into account the reasoning models' higher dollar cost and latency. To this end, we evaluate six state-of-the-art reasoning and non-reasoning LLMs on difficult

questions from the MATH benchmark (Hendrycks et al., 2021b). Our analysis shows that when latency is of no concern, reasoning models already outperform non-reasoning models when the cost of making a mistake is as low $\lambda_E^{\text{critical}} = \$0.01$. When setting the price of latency at \$10/minute (equivalent to human wages of \$600/hour), the critical price of error rises to $\lambda_E^{\text{critical}} = \$10$.

Second, we ask whether LLM cascades (Chen, Zaharia, and Zou, 2023) offer practical benefits, by comparing the performance of a cascade $\mathcal{M}_{\text{small}} \rightarrow \mathcal{M}_{\text{big}}$ against the performance of $\mathcal{M}_{\text{big}}$ by itself—taking into account accuracy, dollar cost, and latency. Surprisingly, we find that $\mathcal{M}_{\text{big}}$ typically outperforms $\mathcal{M}_{\text{small}} \rightarrow \mathcal{M}_{\text{big}}$ for prices of error as low as \$0.1. However, cascade performance notably depends on how well we can quantify the uncertainty of $\mathcal{M}_{\text{small}}$. Interestingly, using Llama3.1 405B as $\mathcal{M}_{\text{small}}$ yields a superior cascade that outperforms $\mathcal{M}_{\text{big}}$ for prices of error up to $\$10,000$, despite the fact that Llama3.1 405B performs comparatively poorly as a standalone model.

To supplement our experiments, we furnish theoretical results on 1) explaining the performance of a cascade $\mathcal{M}_{\text{small}} \rightarrow \mathcal{M}_{\text{big}}$ in terms of a novel covariance-based metric measuring the quality of $\mathcal{M}_{\text{small}}$'s uncertainty signal, and 2) connecting our economic evaluation framework to standard multi-objective optimization based on Pareto optimality.

In summary, our key contributions are the following:

- We propose an economic framework for evaluating LLMs (and LLM systems), which determines a single optimal model based on a use case's economic constraints, all expressed in dollars: the cost of making a mistake (*price of error*), the cost of incremental latency (*price of latency*), and the cost of abstaining from a query (*price of abstention*).
- We state empirical dollar figures for the critical cost of error at which reasoning LLMs offer superior accuracy-cost-latency trade-offs compared to non-reasoning models; and at which a single large LLM $\mathcal{M}_{\text{big}}$ outperforms a cascade $\mathcal{M}_{\text{small}} \rightarrow \mathcal{M}_{\text{big}}$.

Overall, our findings suggest that when automating meaningful human tasks using AI models, accuracy is likely the most important economic factor, outweighing inference costs. Hence, practitioners should typically deploy the most powerful available LLMs.

3.2 Background

Large language models. Large language models (LLM) are transformer-based deep neural networks that auto-regressively generate "tokens" of text, one token at a time (Vaswani et al., 2017). A model $\mathcal{M}$ parametrizes a probability distribution $p_\theta(x_{t+1}|x_1, ..., x_t)$ over the next token given all previous tokens. To generate new text, the user provides a *prompt* $(x_1, ..., x_t)$ and then auto-regressively samples new tokens $x_{t+1}, x_{t+2}, ...$ according to the conditional distribution p_θ. After *post-training* a model's parameters θ using supervised fine-tuning (Wei et al., 2022; Chung et al., 2022) and reinforcement learning (Ouyang et al., 2022), sampling from p_θ returns an appropriate response to the prompt. For example, if the prompt $(x_1, ..., x_t)$ encodes the English sentence "I want to finish all the research articles I start reading. What can I do to improve my self-discipline?", auto-regressive sampling from the post-trained p_θ should yield a list of actionable suggestions.[1]

Systems of large language models. The development of large language models has proven out theoretical scaling laws (Kaplan et al., 2020) predicting that better performance reliably follows from simply implementing bigger models (Brown et al., 2020b; Chowdhery et al., 2022; OpenAI, 2024a). As a result, parameter counts for state-of-the-art LLMs have surged to more than one trillion, requiring significant computing resources for both training and inference. Noting that smaller LLMs often perform well on easier tasks, researchers have proposed system of LLMs in which small and large models collaborate to enhance computational efficiency and reduce inference costs (Chen, Zaharia, and Zou, 2023). For example, *cascades* (Ding et al., 2024; Wang et al., 2024; Narasimhan et al., 2024) delegate queries from small to large LLMs only if the small LLMs are uncertain about the answer, and *routers* (Hari and Thomson, 2023; Hu et al., 2024; Ong et al., 2025) directly send each query to the smallest available model that can still return a satisfactory answer.

Pareto Optimality. Multi-objective optimization (Coello, Aguirre, and Zitzler, 2005; Jin, 2006) is concerned with minimizing a vector-valued function

$$\theta^* = \arg\min \; F(\theta), \tag{3.1}$$

where $F(\theta) = (f_1(\theta), ..., f_k(\theta)) \in \mathbb{R}^k$. Since the distinct objectives $f_i(\theta)$ generally conflict, we define solutions to (3.1) with respect to Pareto optimality (Branke et al.,

[1] You can start today, with this one!

2008). We say that $(f_1(\theta_1), ..., f_k(\theta_1)$ *Pareto-dominates* $(f_1(\theta_2), ..., f_k(\theta_2)$, or

$$(f_1(\theta_1), ..., f_k(\theta_1) <_{\mathrm{P}} (f_1(\theta_2), ..., f_k(\theta_2), \tag{3.2}$$

if $f_i(\theta_1) \leq f_i(\theta_2)$ for all $i = 1, 2, ..., k$ and at least one of the inequalities is strict. We consider θ^* to be a solution of (3.1) if there exists no θ' such that $(f_1(\theta'), ..., f_k(\theta') <_{\mathrm{P}} (f_1(\theta^*), ..., f_k(\theta^*)$. The set of such solutions to (3.1) makes up the *Pareto frontier*

$$\mathcal{P} = \{\theta \in \Theta \mid \forall \theta' \in \Theta,\ F(\theta') \not<_P F(\theta)\}. \tag{3.3}$$

We interchangeably refer to the image $F(\mathcal{P}) = \{(f_1(\theta), ..., f_k(\theta)) \mid \theta \in \mathcal{P}\}$ as the Pareto frontier. The multi-objective performance of LLMs and LLM systems is typically evaluated by computing the Pareto frontier of the performance metrics $f_1(\theta) :=$ error rate and $f_2(\theta) :=$ cost.

When comparing the performance of individual LLMs, θ represents the model's identity (for example, GPT 4.1 or DeepSeek R1) and other hyperparameters (user and system prompts, sampling configuration, etc.). By contrast, for systems of LLMs, θ typically denotes the system's operating point, e.g., the deferral rate or confidence threshold for an LLM cascade $\mathcal{M}_{\text{small}} \to \mathcal{M}_{\text{big}}$.

3.3 Economic Evaluation of LLMs

In this section, we describe our economic framework for evaluating the multi-objective performance trade-offs of LLMs (and systems of LLMs).

LLMs as Agents

Using the language of reinforcement learning, we cast LLMs and systems of LLMs as *agents* that reap per-query *rewards* from their chosen *actions*. As we will show, this methodology provides a natural basis for multi-objective optimization of LLM performance from an economic perspective. Rather than present our framework in the abstract, we illustrate our formalism for three concrete LLM systems: standalone LLMs, cascades (Chen, Zaharia, and Zou, 2023), and routers (Shnitzer et al., 2023).

Standalone LLM. An LLM's action space consists of its possible text generations:

$$\mathcal{A} = \{y \mid y \in \Sigma^*\}, \tag{3.4}$$

where Σ is the alphabet of tokens (Kudo and Richardson, 2018). Following other authors, to evaluate the quality of the output y we assume the existence of a binary

error-calling mechanism $s(y) : \Sigma^* \to \{0, 1\}$ that maps each string output to a judgment of whether the output is "satisfactory."[2] We refer to each y with $s(y) = 1$ as an *error* and use the notation $\mathbb{1}_E(y)$ as a shorthand for the indicator $\mathbb{1}[s(y) = 1]$.

For each action $y \in \mathcal{A}$, the LLM reaps a reward

$$r = -\left(C + \lambda_L L + \lambda_E \mathbb{1}_E\right), \tag{3.5}$$

where C is the dollar cost of generating the output y, and L is the latency.

LLM Cascade. A *cascade* $C = M_1 \to M_2 \to ... \to M_k$ is a system of LLMs M_1, M_2, ..., M_k that passes each incoming query from M_i to M_{i+1} until it encounters a model M_τ $(1 \le \tau \le k)$ with sufficient confidence to answer the query.

We think of the entire cascade as an agent with action space

$$\mathcal{A} = \{(\tau, y) \mid 1 \le \tau \le k, y \in \Sigma^*\}, \tag{3.6}$$

where y denotes the cascade's output, and τ represents the index of the model M_τ responsible for the cascade's output y. Specifically, τ is the index of the first model that does not defer the query.

As for a standalone LLM, the cascade reaps a reward $r = -\left(C + \lambda_L L + \lambda_E \mathbb{1}_E\right)$ for each action (τ, y). However, the cost and latency depend on τ:

$$C = \sum_{j=1}^{\tau} C_j, \qquad L = \sum_{j=1}^{\tau} L_j, \tag{3.7}$$

where C_j and L_j are the cost and latency of model M_j on the query. These equations result from the "cascading" nature of a cascade: we pay for each successive model—both in dollar cost and in latency—until we reach the earliest model with sufficient confidence, M_τ.

LLM Router with Abstention. A *router with abstention* is a function $g(x) : \Sigma^* \to \{1, 2, ..., k\} \cup \{\varnothing\}$ that routes each incoming query x to one of k LLMs M_1, ..., M_k in a single step, or abstains from answering the query ($\varnothing$). Its action space is

$$\mathcal{A} = \{(i, y) \mid i \in \{1, 2, ..., k\}, y \in \Sigma^*\} \cup \{\varnothing\}, \tag{3.8}$$

where (i, y) denotes that model i generates output y, and $\varnothing$ indicates that the router abstained (for example, to defer the query to a human expert). Analogous to a cascade, the router with abstention reaps a reward $r = -\left[C + \lambda_L L + \lambda_E \mathbb{1}_E(y) + \lambda_A \mathbb{1}_A(y)\right]$

[2] Optionally, the error-calling function s may take into account a reference answer y_{ref}, leading to a bivariate function $s(y, y_{\text{ref}}) : \Sigma^* \times \Sigma^* \to \{0, 1\}$.

for each action, the sole difference being the addition of a term $\mathbb{1}_A$ indicating that the router abstained. However, the cost C and latency L are computed differently:

$$C = c_0 + C_i, \qquad L = l_0 + L_i, \tag{3.9}$$

where C_i and L_i are the cost and latency of generating the output with model M_i. The constants c_0 and l_0 represent the computational overhead of determining the routing decision $g(x) \in \{1, 2, ..., k\}$. These costs are typically negligible, as g is usually lightweight compared to the LLMs M_1, M_2, ..., M_k.[3]

Economic Modeling of LLM Use Cases

To cast standalone LLMs, cascades, and routers as reward-maximizing agents, we defined the per-query reward

$$r = -\left[C + \lambda_L L + \lambda_E \mathbb{1}_E + \lambda_A \mathbb{1}_A\right], \tag{3.10}$$

where C is the dollar cost of processing a query, L is the latency, $\mathbb{1}_E$ indicates an error, and $\mathbb{1}_A$ indicates an abstention. In general, we formulate this reward as

$$r = -\left[C + \sum_{\mu \in \mathcal{P}_{\text{numeric}}} \lambda_\mu \, \mu \; + \sum_{\chi \in \mathcal{P}_{\text{binary}}} \lambda_\chi \mathbb{1}_\chi\right], \tag{3.11}$$

where $\mathcal{P}_{\text{numeric}} \cup \mathcal{P}_{\text{binary}}$ is the set of per-query performance metrics: $\mathcal{P}_{numeric}$ represents the numeric performance metrics (for example, latency) and $\mathcal{P}_{binary}$ denotes the binary performance events (for example, error and abstention). This definition includes other reasonable performance objectives, such as privacy (Zhang et al., 2024a).

The coefficients $\{\lambda_\mu\}_{\mu \in \mathcal{P}_{\text{numeric}}}, \{\lambda_\chi\}_{\chi \in \mathcal{P}_{\text{binary}}} \in \mathbb{R}^+$ are *prices* measuring the economic impact when the performance metrics worsen. Table 3.1 gives a few key examples:

These parameters are based on the economic concept of *indifference* (Mankiw, 2020). For example, the price of error λ_E is the lowest dollar figure d at which the user (i.e., the organization that deploys the LLM system) would be indifferent between suffering an LLM error or receiving d dollars in cash.

[3]Typically, the routing model takes the form of a deep neural network—for example, a finetuned small language model—with less than 1B parameters (Shnitzer et al., 2023; Hari and Thomson, 2023).

Table 3.1: Examples of key economic parameters in our framework.

Parameter	Symbol	Definition	Units
Price of Error	λ_E	Amount the user is willing to pay to avoid a single prediction error.	\$
Price of Latency	λ_L	Amount the user is willing to pay to reduce per-query latency by one second.	\$/sec
Price of Abstention	λ_A	Amount the user is willing to pay to avoid a model abstention (no answer).	\$

Multi-Objective Performance Evaluation in a Single Number

Given the user's economic constraints (see Table 3.1), the expected per-query reward of the LLM system is

$$R(\lambda; \theta) = \mathbb{E}_\theta[r(\lambda)], \tag{3.12}$$

where r is the per-query reward (3.11). We use λ to denote the totality of economic parameters (e.g., λ_E, λ_L, ...), and denote the configuration of the LLM system by θ. Selecting the optimal LLM for a given use case, or optimizing the operating point of a system of LLMs, involves the reward maximization

$$\theta^*(\lambda) = \arg\max_\theta R(\lambda; \theta). \tag{3.13}$$

When choosing among individual LLMs, θ represents the identity of the model (e.g., GPT 4.1 vs DeepSeek R1), as well as hyperparameter settings (user and system prompts, sampling temperature, top-p, top-k, etc.). On the other hand, for LLM systems, θ typically represents tunable parameters such as the confidence thresholds of LLM cascades (Zellinger and Thomson, 2025b).

Often the user's economic constraints λ are not known with certainty. In this case, it is instructive to compute optimal models for a range of potential λ values. These sensitivity tables can be highly informative, as the optimal model is often stable over a wide range of different economic constraints. See sections 3.4 and 3.4 for examples.

To compare the performance of different LLMs (or systems of LLMs) across different economic scenarios λ, we consider their expected per-query rewards for the optimal choices of θ:

$$R(\lambda) = R(\lambda; \theta^*(\lambda)). \tag{3.14}$$

Alternatively, if λ is uncertain—for example, suppose that new legislation may change the expected payout of medical malpractice lawsuits, potentially raising the

price of error λ_E for medical AI deployments—we model λ as a random variable $\lambda \sim p(\lambda)$, yielding the expected per-query reward

$$R = \mathbb{E}_{\lambda \sim p(\lambda)}[R(\lambda)]. \tag{3.15}$$

Estimating the Price of Error

Managing the potential cost of LLM mistakes is critical for businesses, especially those in risk-sensitive industries (e.g., finance, law, or medicine).

In this section, we illustrate how to estimate the price of error by walking through an example calculation for medical diagnosis. Practitioners may then adapt these steps to their own industries.

Estimate of λ_E for medical diagnosis. We estimate the price of error for medical diagnosis to be about \$1,000. We arrive at this number by considering data on medical malpractice lawsuits and applying Bayes' theorem.

For a single diagnosis, denote the event of a medical malpractice lawsuit as M and the event of a diagnostic error as E. Our estimate for the price of error is then

$$\hat{\lambda}_E = \mathbb{E}[\text{Cost}|M] \times \mathbb{P}(M|E) \tag{3.16}$$

$$= \mathbb{E}[\text{Cost}|M] \times \frac{\mathbb{P}(E|M)\mathbb{P}(M)}{\mathbb{P}(E)}. \tag{3.17}$$

Studdert et al. (2006) report that the mean payout for medical malpractice lawsuits is \$485,348, so we use $\mathbb{E}[\text{Cost}|M] = \$500,000$. In addition, two thirds of malpractice suits are derived from a genuine medical error (rather than a fraudulent claim), so $\mathbb{P}(E|M) = 2/3$.

Jena et al. (2011) estimate a doctor's yearly risk of facing a malpractice claim as 7.4%, so we estimate that a doctor encounters a malpractice suit once every 1/0.074 = 13.5 years. Assuming the doctor makes 3 diagnoses per hour, we arrive at around 100,000 diagnoses within this time frame, so $\mathbb{P}(M) \approx 1/100,000$. By contrast, Singh, Meyer, and Thomas (2014) estimate that 1 in 20 adults experiences a diagnostic error each year. Taking into account the fact that people may go to the hospital more than once per year, and that each visit may involve more than one diagnosis, we use $\mathbb{P}(E) \approx 1/100$.

Plugging these numbers into the formula (3.17), we arrive at the estimate $\boxed{\hat{\lambda}_E \approx \$333}$ for medical diagnosis.

Connections between Economic Evaluation and Pareto Optimality

In this section, we establish theoretical connections between our economic evaluation framework and Pareto optimality. Our first result shows that sweeping over different economic scenarios λ recovers the full Pareto frontier for the performance metrics, assuming regularity conditions.

Theorem 3. *Let $\theta^*(\lambda)$ be the solution to the reward maximization problem*

$$\theta^* = argmax_\theta \quad R(\lambda; \theta), \tag{3.18}$$

where $\theta \in \mathbb{R}^p$ denotes an LLM system's tunable parameters, and λ is the vector of economic costs as defined in Section 3.3. Assume that regularity conditions hold, such that for each $\lambda \in \mathbb{R}_{>0}^{|\mathcal{P}_{numeric}|+|\mathcal{P}_{binary}|}$ there exist bounds $\{\gamma_\mu\}_{\mu\in\mathcal{P}_{numeric}}$ and $\{\gamma_\chi\}_{\mu\in\mathcal{P}_{binary}} > 0$ such that $\theta^*(\lambda)$ is equivalently the solution of the constrained optimization problem

$$\begin{aligned} \theta^* = argmin_\theta \quad & \hat{\mathbb{E}}_\theta[C] \\ subject\ to \quad & \hat{\mathbb{E}}_\theta[\mu] \le \gamma_\mu, \ \ \mu \in \mathcal{P}_{numeric} \\ & \hat{\mathbb{E}}_\theta[\mathbb{1}_\chi] \le \gamma_\chi, \ \ \chi \in \mathcal{P}_{binary}, \end{aligned} \tag{3.19}$$

and vice versa for $\gamma \mapsto \lambda(\gamma)$. Then the vector of economic costs, λ, maps surjectively onto the Pareto surface via the mapping

$$\lambda \mapsto (\hat{\mathbb{E}}_{\theta^*(\lambda)}[C], \hat{\mathbb{E}}_{\theta^*(\lambda)}[\mu_1], ..., \hat{\mathbb{E}}_{\theta^*(\lambda)}[\mu_{|\mathcal{P}_{numeric}|}], \hat{\mathbb{P}}_{\theta^*(\lambda)}[\chi_1], ..., \hat{\mathbb{P}}_{\theta^*(\lambda)}[\chi_{|\mathcal{P}_{binary}|}]). \tag{3.20}$$

Proof. See Appendix B. □

The next result provides theoretical support for evaluating the overall performance of LLM systems by comparing the expected reward (3.14) across a grid of possible use cases λ.

Theorem 4. *Consider two LLM systems, $\mathcal{S}_1$ and $\mathcal{S}_2$. Assume that the regularity assumptions of Theorem 3 hold. If the expected rewards (3.14) satisfy*

$$R_1(\lambda) \ge R_2(\lambda)$$

for all $\lambda \in \mathbb{R}_{>0}^{|\mathcal{P}_{numeric}|+|\mathcal{P}_{binary}|}$, then no point on the Pareto surface for $\mathcal{S}_1$ dominates any point on the Pareto surface for $\mathcal{S}_2$.

Proof. Suppose for the sake of contradiction that θ_2 on the Pareto surface of $\mathcal{S}_2$ dominates θ_1 on the Pareto surface of $\mathcal{S}_1$. By Theorem 3, there exist $\lambda_1, \lambda_2 \in \mathbb{R}^+$ such that

$$\theta_1 = \theta^*(\lambda_1), \tag{3.21}$$

$$\theta_2 = \theta^*(\lambda_2), \tag{3.22}$$

as defined by (3.13). Hence, we have

$$R_1(\lambda_1) = R_1(\lambda_1; \theta^*(\lambda_1)) = R(\lambda_1; \theta_1) < R_2(\lambda_1; \theta_2) \leq \max_{\theta} R_2(\lambda_1; \theta) = R_2(\lambda_1), \tag{3.23}$$

where the middle inequality follows from the assumed Pareto dominance of θ_2 over θ_1. □

3.4 Experiments

We apply our economic evaluation framework to explore the practical relevance of less powerful LLMs. We address two concrete questions:

- When do reasoning models outperform non-reasoning models? (Section 3.4)
- When does a single large model $\mathcal{M}_{\text{big}}$ outperform a cascade $\mathcal{M}_{\text{small}} \to \mathcal{M}_{\text{big}}$? (Section 3.4)

Methodology

Since AI models are increasingly considered as a replacement for meaningful human labor, we focus our analysis on difficult mathematics questions from the MATH benchmark (Hendrycks et al., 2021b). This benchmark offers the advantage of containing ground truth difficulty labels (1-5) as well as reference answers. To simplify evaluation of LLM answers, we filter out questions with non-numeric answers and use stratified sampling to obtain 500 questions for each of the three difficulty levels 1, 3, and 5. We exclusively use the training split of the MATH benchmark; although this choice heightens the potential for data contamination (Ravaut et al., 2025), it makes available a greater number of difficult examples and therefore improves the statistical power of our experiments.

Models. We evaluate six state-of-the-art LLMs (three reasoning, three non-reasoning): Meta's Llama3.3 70B and Llama3.1 405B models (Meta AI, 2024), OpenAI's GPT4.1 and o3 models, DeepSeek's R1 model (DeepSeek AI, 2025), and Alibaba's

Qwen3 235B-A22 model (Alibaba AI, 2025). We prompt each model using zero-shot chain-of-thought (Kojima et al., 2022).

LLM cascades. To evaluate cascades $\mathcal{M}_{\text{small}} \rightarrow \mathcal{M}_{\text{big}}$, we quantify the small model's probability of correctness using self-verification, also known as *P(True)* (Kadavath et al., 2022; Zellinger and Thomson, 2025b). When comparing the performance of $\mathcal{M}_{\text{small}} \rightarrow \mathcal{M}_{\text{big}}$ against that of $\mathcal{M}_{\text{big}}$ in Section 3.4, we split the data into 50% training and test sets; we use the training set exclusively for estimating the optimal confidence threshold, and evaluate cascade performance on the test split ($n = 250$).

Metrics. We measure the correctness, dollar cost, and latency for each query by invoking LLMs via the commercial Fireworks AI (Llama3.1, Llama3.3, Qwen3, DeepSeek R1) and OpenAI (GPT 4.1, o3) application programming interfaces.

Correctness. To assess correctness of a model's answer, we invoke Llama3.1 405B with an evaluation prompt containing the ground truth reference answer.

Cost. We compute

$$C = N_{\text{in}} \times C_{\text{in}} + N_{\text{out}} \times C_{\text{out}}, \tag{3.24}$$

where $N_{\text{in}}, N_{\text{out}}$ are the numbers of input and output tokens, and $C_{\text{in}}, C_{\text{out}}$ are the API providers' model-specific prices, expressed in dollars per token.

Latency. We record the time before and after an API call to the LLM model provider. Hence, our reported latencies include internet roundtrip latency. However, this additional latency (<300ms) is negligible, being 10-200x smaller than the latencies we observe for answering queries.

Cascade Error Reduction: To predict the performance of a cascade $\mathcal{M}_{\text{small}} \rightarrow \mathcal{M}_{\text{big}}$ based on the quality of the confidence signal of $\mathcal{M}_{\text{small}}$ (self-verification in our case), we introduce the *cascade error reduction*

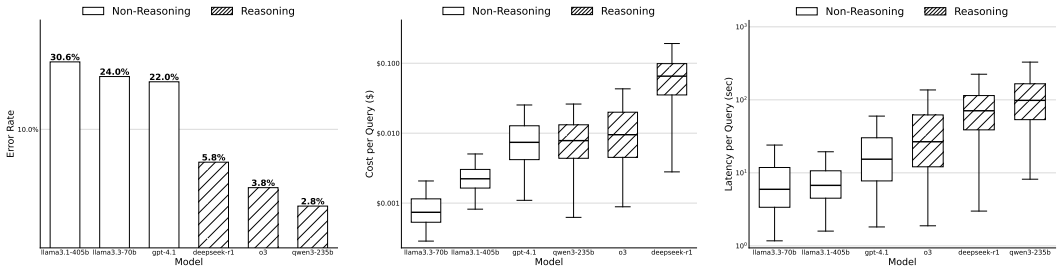
$$\text{CER} = \text{Cov}(\mathbb{1}_D, \mathbb{1}_{\text{error}}^{\mathcal{M}_{\text{small}}}), \tag{3.25}$$

where $\mathbb{1}_D$ indicates the small model's decision to defer the query to $\mathcal{M}_{\text{big}}$, and $\mathbb{1}_{\text{error}}^{\mathcal{M}_{\text{small}}}$ indicates that the output of $\mathcal{M}_{\text{small}}$ is incorrect. We theoretically justify this metric in Theorem 5.

See appendices C-E for more details on methodology.

Baseline Performance: Error Rate, Cost, and Latency

Figure 3.2 shows the performance of reasoning and non-reasoning models on $n = 500$ of the most difficult questions of the MATH benchmark. Clearly, reasoning models have much lower error rates. However, their costs per query are 10-100x greater, and latencies per query are up to 10x greater.



(a) Error (b) Cost per Query ($) (c) Latency per Query (sec)

Figure 3.2: On the most difficult questions of the MATH benchmark, reasoning models have much lower error rates but are 10-100x more expensive and take 10x longer to answer a query.

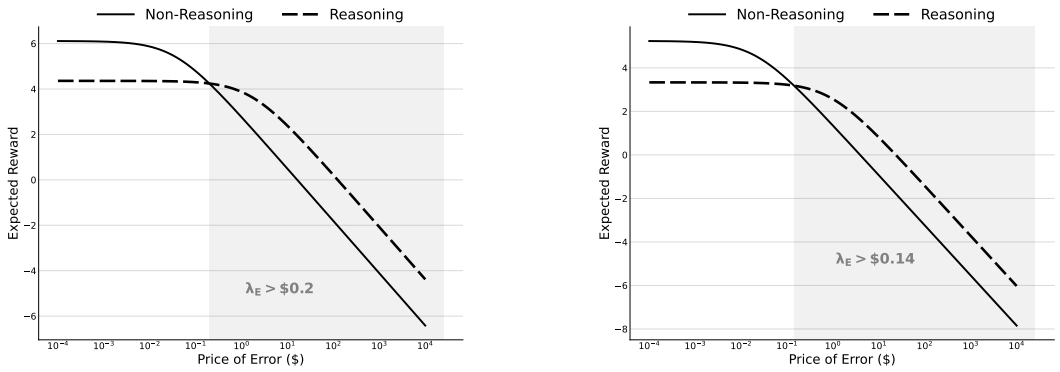
Reasoning models are known to dynamically scale the number of output tokens based on the difficulty of the query. However, non-reasoning models prompted with chain-of-thought also adapt the number of output tokens based on query difficulty. Figure .14 in Appendix A compares the number of output tokens across queries of varying difficulty, showing that the relative increase in output tokens is comparable between reasoning and non-reasoning models; however, reasoning models have higher baseline numbers of output tokens.

When Do Reasoning Models Outperform Non-Reasoning Models?

We now leverage our economic evaluation framework to determine under what conditions reasoning models outperform non-reasoning models on difficult questions from MATH.

First, we only trade-off accuracy and cost, disregarding latency. Second, we simultaneously consider accuracy, cost, and latency, and display the optimal LLM over a range of economic constraints.

Figure 3.3 plots the expected reward (3.14) of reasoning and non-reasoning LLMs on MATH, for prices of error ranging from $\lambda_E = \$0.0001$ to $\lambda_E = \$10,000$ per query. These curves only trade-off accuracy and cost, assuming that latency costs nothing ($\lambda_L = \$0/\text{sec}$). Each curve shows the average expected reward across all models of one category (reasoning or non-reasoning).



(a) Difficulty: Level 3

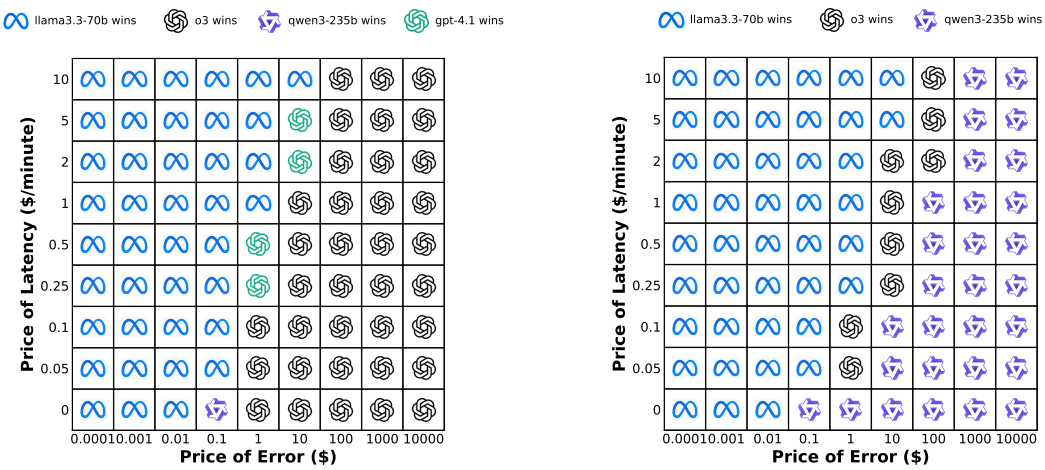
(b) Difficulty: Level 5

Figure 3.3: Reasoning models offer superior accuracy-cost trade-offs as soon as the price of error, λ_E, exceeds \$0.20 per query. The y-axis shows -log(-$R(\lambda_E)$) to make the trends more easily visible.

The results show that reasoning models offer superior accuracy-cost trade-offs as soon as the cost of making a mistake exceeds \$0.20. This figure is surprisingly low. Suppose a human worker is able to complete the same task in 5 minutes on average, and assume that the consequence of a mistake is having to re-do the task. Then the economic loss of a single mistake exceeds \$0.20 as soon as the worker's wages exceed \$2/hour—a number below the U.S. federal minimum wage.

Figure 3.4 introduces a non-zero price of latency and displays the optimal models across economic constraints, with prices of error λ_E ranging from \$0.0001 to \$10,000, and prices of latency λ_L ranging from \$0/minute to \$10/minute. These prices of latency correspond to human wages from \$0/hour to \$600/hour. Hence, we believe this range captures a wide variety of use cases for automating meaningful human tasks, ranging from customer support (below \$100/hour) to medical diagnosis (above \$100/hour). We note that this regime does **not** include the more stringent latency constraints of using LLMs for non-human tasks such as serving popular web applications (Kohavi and Longbotham, 2007) or iterating through a large number of database records. We leave exploration of such tasks—and their higher prices of latency—to future work, as we are most concerned with the emerging practice of automating human tasks using LLMs.

Figure 3.4 shows that reasoning models generally outperform non-reasoning models for prices of error above \$10 when the price of latency is at most \$5/minute (equivalent to human wages of \$300/hour). For a price of latency of \$10/minute (or \$600/hour), the critical price of error rises to \$100. Among LLMs, Qwen3-235B-A22B, o3, and Llama3.3 70B emerge as the preferred models for the vast majority



(a) Difficulty: Level 3

(b) Difficulty: Level 5

Figure 3.4: Optimal models for different combinations of *price of error* and *price of latency*. Reasoning models show superior performance for prices of error above \$10.

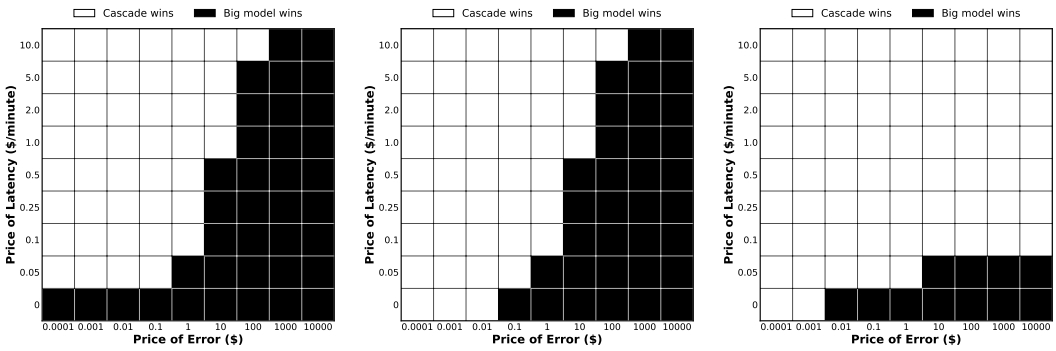
of economic scenarios.

When Does a Single Big Model Outperform a Cascade?

A large language model cascade $\mathcal{M}_{\text{small}} \rightarrow \mathcal{M}_{\text{big}}$ sends all queries first to a relatively small model $\mathcal{M}_{\text{small}}$. If $\mathcal{M}_{\text{small}}$ is uncertain about the answer, it defers the query to $\mathcal{M}_{\text{big}}$; otherwise, it directly returns the output. Cascading generally assumes that $\mathcal{M}_{\text{big}}$ performs better than $\mathcal{M}_{\text{small}}$ on all queries; hence, we expect that as the price of error λ_E increases, there exists a cross-over point $\lambda_E^{\text{critical}}$ when directly sending queries to the big model $\mathcal{M}_{\text{big}}$ outperforms cascading.

Figure 3.5 compares the performance of three different cascades $\mathcal{M}_{\text{small}} \rightarrow \mathcal{M}_{\text{big}}$. For each cascade, $\mathcal{M}_{\text{big}}$ = Qwen3 235B-A22B, but $\mathcal{M}_{\text{small}}$ ranges over all the non-reasoning models (Llama3.3 70B, Llama3.1 405B, and GPT-4.1). We evaluate only on the most difficult (level 5) questions of the MATH benchmark. We tune the cascade's deferral threshold on $n = 250$ training examples and use the remaining $n = 250$ questions for evaluation.

Disregarding the impact of latency, using $\mathcal{M}_{\text{big}}$ = Qwen3 235B-A22 as a standalone LLM generally outperforms the cascade $\mathcal{M}_{\text{small}} \rightarrow \mathcal{M}_{\text{big}}$—as soon as the price of error exceeds $\lambda_E > \$0.10$. As the price of latency increases to $\lambda_L = \$0.5/\text{minute}$ (equiv. to \$30/hour), the critical price of error $\lambda_E^{\text{critical}}$ ratchets up to \$10. At $\lambda_L = \$10/\text{minute}$ (equiv. to \$600/hour), $\lambda_E^{\text{critical}}$ increases to $\$1,000$. This finding



(a) GPT 4.1 $\rightarrow \mathcal{M}_{\text{big}}$ (b) Llama3.3 70B $\rightarrow \mathcal{M}_{\text{big}}$ (c) Llama3.3 405B $\rightarrow \mathcal{M}_{\text{big}}$

Figure 3.5: Directly sending queries to Qwen3 235B-A22B ($\mathcal{M}_{\text{big}}$) outperforms cascades when the cost of making a mistake exceeds \$0.10 and the price of latency is sufficiently low.

suggests that when automating medical diagnosis (with an estimated price of error between \$100 and \$1,000, as discussed in Section 3.3), it may be preferable to avoid cascading.

However, this analysis carries an important caveat. For the cascade with $\mathcal{M}_{\text{small}}$ = Llama3.3 405B, we observed a marked and surprising outperformance over the other cascades—even though as a standalone LLM, Llama3.3 405B performs strictly worse than Llama3.3 70B, as shown in Figure 3.2. The cascade Llama3.1 405B $\rightarrow$ Qwen3 235B-A22B yields better accuracy-cost-latency trade-offs than Qwen3 235B-A22B for the vast majority of economic scenarios, including prices of error up to \$10, 000 and prices of latency up to \$10/minute (equiv. to \$600/hour).

Why does using Llama3.1 405B as the small model result in superior cascade performance? To explain this phenomenon, we point to the model's remarkable self-verification performance. To lay out our argument, we first restate the formula for the error rate of a two-model cascade given without proof by Zellinger and Thomson (2024):

Theorem 5 (Cascade Error)**.** *Consider a cascade $\mathcal{M}_{small} \rightarrow \mathcal{M}_{big}$, where the deferral decision of $\mathcal{M}_{small}$ is determined by the indicator $\mathbb{1}_D$. Then the error rate, $e_{cascade}$ of the cascade is*

$$e_{\mathcal{M}_{small} \rightarrow \mathcal{M}_{big}} = (1 - p_d)\, e_{\mathcal{M}_{small}} + p_d\, e_{\mathcal{M}_{big}} + Cov(\mathbb{1}_D, \mathbb{1}_{error}^{\mathcal{M}_{big}}) - Cov(\mathbb{1}_D, \mathbb{1}_{error}^{\mathcal{M}_{small}}), \tag{3.26}$$

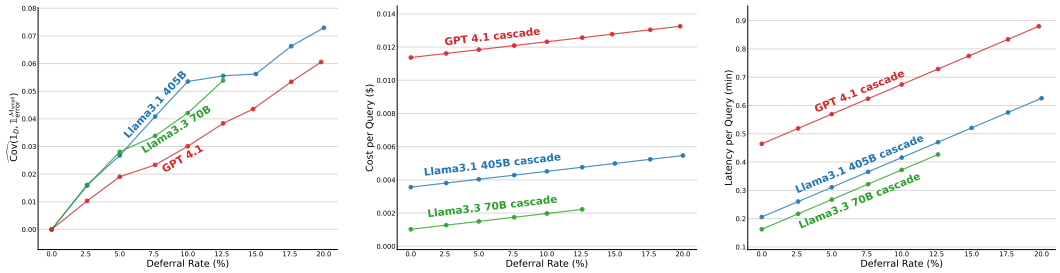
where $p_d := \mathbb{E}[\mathbb{1}_D]$ is the deferral rate, $e_{\mathcal{M}_{small}} := \mathbb{E}[\mathbb{1}_{error}^{\mathcal{M}_{small}}]$ is the error rate of $\mathcal{M}_{small}$, and $e_{\mathcal{M}_{big}} := \mathbb{E}[\mathbb{1}_{error}^{\mathcal{M}_{big}}]$ is the error rate of $\mathcal{M}_{big}$.

Proof. This result follows from writing out the formula for cascade error,

$$e_{\mathcal{M}_{\text{small}} \to \mathcal{M}_{\text{big}}} = \mathbb{E}[(1 - \mathbb{1}_D)\mathbb{1}_{\text{error}}^{\mathcal{M}_{\text{small}}} + \mathbb{1}_D \mathbb{1}_{\text{error}}^{\mathcal{M}_{\text{big}}}]. \tag{3.27}$$

Using the linearity of expectation, followed by adding and subtracting terms to recover the covariances, gives the result. □

Theorem 5 shows that the error rate of a cascade $\mathcal{M}_{\text{small}} \to \mathcal{M}_{\text{big}}$, relative to randomly sending queries to $\mathcal{M}_{\text{small}}$ and $\mathcal{M}_{\text{big}}$, depends on the difference in covariances $\text{Cov}(\mathbb{1}_D, \mathbb{1}_{\text{error}}^{\mathcal{M}_{\text{big}}}) - \text{Cov}(\mathbb{1}_D, \mathbb{1}_{\text{error}}^{\mathcal{M}_{\text{small}}})$. Intuitively, these covariances measure the increase in the models' error rates conditional on deferral. Specifically, $\text{Cov}(\mathbb{1}_D, \mathbb{1}_{\text{error}}^{\mathcal{M}_{\text{small}}})$ expresses the agreement between the deferral decision and the true uncertainty of $\mathcal{M}_{\text{small}}$.[4] In practice, $\text{Cov}(\mathbb{1}_D, \mathbb{1}_{\text{error}}^{\mathcal{M}_{\text{big}}}) \ll \text{Cov}(\mathbb{1}_D, \mathbb{1}_{\text{error}}^{\mathcal{M}_{\text{small}}})$, so $\text{Cov}(\mathbb{1}_D, \mathbb{1}_{\text{error}}^{\mathcal{M}_{\text{small}}})$ alone is a strong indicator of cascade effectiveness. We refer to this metric as the *cascade error reduction* (CER).



(a) CER (b) Cost (c) Latency

Figure 3.6: Using Llama3.1 405B as $\mathcal{M}_{\text{small}}$ yields superior cascading performance, despite its subpar performance as a standalone model, because it performs better at self-verification—indicated by a higher cascade error reduction (CER) (a). In contrast to error rate, cost and latency are simple linear functions of the deferral rate (b and c).

Figure 3.6 plots CER = $\widehat{\text{Cov}}(\mathbb{1}_D, \mathbb{1}_{\text{error}}^{\mathcal{M}_{\text{small}}})$ against the cascades' deferral thresholds. The figure shows that $\widehat{\text{Cov}}(\mathbb{1}_D, \mathbb{1}_{\text{error}}^{\mathcal{M}_{\text{small}}})$ is highest for Llama3.1 405B, explaining its superior suitability for playing the role of $\mathcal{M}_{\text{small}}$ in a cascade (Figure 3.5). Figures 3.6b and 3.6c plot the cascades' costs and latencies for the same range of deferral rates, verifying that Llama3.1 405B's superior cascade performance is driven by error rate, not cost or latency.

[4] Thus, $\text{Cov}(\mathbb{1}_D, \mathbb{1}_{\text{error}}^{\mathcal{M}_{\text{small}}})$ is closely related to the area under the accuracy-rejection curve (AUARC) from selective prediction (El-Yaniv and Wiener, 2010).

3.5 Related Work

In this section, we discuss related work.

Economic Analysis of LLMs. Several papers have explored the economic and labor market impacts of large language models (Brynjolfsson, Li, and Raymond, 2023; Eloundou et al., 2024). For example, Eloundou et al. (2024) assess the vulnerability of different jobs to AI automation, finding that highly paid professional work is particularly exposed. Different from our work, the authors define exposure with respect to achieving time savings while maintaining the same quality of work. By contrast, we wish to highlight the comparatively low cost of AI labor, as reflected in costs per query between \$0.01 and \$0.1, even for state-of-the-art reasoning LLMs. Our argument is that lowering the cost of labor—even without saving time—yields significant economic benefits.

When it comes to ranking the performance of LLMs based on economic principles, the closest work to ours is the preprint by Erol et al. (2025). The authors propose quantifying accuracy-cost trade-offs by the ratio of cost to accuracy. Numerically, this approach is mathematically similar to our multi-objective reward (3.10) when considering only accuracy and cost, and using a constant, unchanging price of error for all use cases. We believe that our framework's ability to handle a greater number of simultaneous performance objectives, and to differentiate between the economic realities of different industries (for example, customer support vs medical diagnosis), to be significant strengths.

Moreover, we consider the theoretical motivation of Erol et al. (2025)'s cost-of-pass metric to be flawed. Their argument is that with an accuracy of a, an LLM can be viewed to generate the correct answer to a query in $1/a$ attempts (assuming a geometric distribution). Hence, their metric measures the total cost ("cost of pass") required to produce the correct answer. However, the correctness of repeated samples from an LLM with accuracy a may *only* be modeled as i.i.d. Bernoulli(a) trials if we also randomly sample new queries q. For a fixed query, repeated sampling does not reliably yield a correct answer, although it offers some benefits (Chen et al., 2024b).

LLM Evaluation and Benchmarking. The evaluation of large language models has received significant attention over the past few years (Laskar et al., 2024). Prior work has mostly focused on assessing specific capabilities of LLMs, such as summarization (Narayan, Cohen, and Lapata, 2018), general knowledge (Hendrycks and Gimpel, 2018), truthfulness (Lin, Hilton, and Evans, 2022b), mathematical

reasoning (Hendrycks et al., 2021b), and others. In general, individual benchmarks rapidly become obsolete as LLMs become more capable. Notably, Liang et al. (2023) provide a compelling synthesis of many of the different capabilities worth evaluating. In addition to benchmarking, the Chatbot Arena (Chiang et al., 2024) has popularized ranking LLMs through crowd-sourced pairwise comparisons, in a manner similar to reinforcement learning from human feedback (Ouyang et al., 2022).

These evaluation efforts are not directly comparable to our work, as we focus on *multi-objective* evaluation of conflicting performance objectives, such as accuracy, cost, and latency. It is not sufficient to evaluate these metrics individually, since we wish to simultaneously optimize for these performance objectives.

Multi-Objective LLM Systems. Our emphasis on multi-objective evaluation is reflected in research on multi-LLM systems such as cascades (Ding et al., 2024; Chen, Zaharia, and Zou, 2023; Aggarwal et al., 2024) and routers (Shnitzer et al., 2023; Hari and Thomson, 2023; Jitkrittum et al., 2025). Researchers have mainly framed the accuracy-cost trade-off as a constrained minimization of the error rate subject to a cost budget (Chen, Zaharia, and Zou, 2023; Jitkrittum et al., 2024; Hu et al., 2024). Evaluation of different LLM systems proceeds by plotting Pareto-optimal error rates against the corresponding cost budgets. Unfortunately, this approach does not easily generalize to more than two performance objectives, since it is difficult to compare the quality of higher-dimensional Pareto frontiers (Zellinger, Liu, and Thomson, 2025). Specifically, the volume under a Pareto surface is not a meaningful metric unless the lower-dimensional projections of different Pareto frontiers substantially intersect.[5]

In addition, we suspect that prior work's emphasis on setting budgets for cost or latency may be overly influenced by the peculiarities of the IT industry, as these budgets closely reflect that industry's *service-level objectives* (SLO). However, artificial intelligence is a society-wide phenomenon that extends far beyond IT. As AI starts to perform meaningful human work, it will likely emerge as a revenue generator rather than a cost center—diminishing the relevance of cost budgets as a mental framework.

Computation of Pareto Frontiers. From an optimization perspective, our method-

[5]The same problem may occur in two dimensions. Consider error-cost curves that are horizontally shifted, i.e., the attainable cost budgets are disjoint. Comparing the areas under these curves does not yield meaningful results.

ology corresponds to a well-known method for approximating Pareto frontiers called the *weighted sum method* for *scalarization* (Koski, 1988; Jahn, Klose, and Merkel, 1991; Banholzer and Volkwein, 2019). Prior work has identified certain disadvantages of scalarization: for example, it may not yield all Pareto-optimal trade-offs when the Pareto surface is non-convex, or coverage of the Pareto frontier might be unevenly distributed (Das and Dennis, 1997). Such issues are not a major concern for us, as we motivate our methodology on economic grounds, by casting LLM systems as reward-maximizing agents (see Section 3.3). We note, however, that Pareto frontiers arising in performance evaluation of LLM systems tend to be convex since randomly routing queries between two LLMs smoothly interpolates between their respective performance metrics.

3.6 Conclusion

We have presented an economic framework for evaluating the performance of LLMs and LLM systems. Compared to plotting Pareto frontiers, our approach yields a single optimal model based on a use case's economic constraints: the cost of making a mistake (*price of error*), the cost of incremental latency (*price of latency*), and the cost of abstaining from a query (*price of abstention*), as well as possible additional objectives such as privacy (Zhang et al., 2024a).

We motivated our framework by casting LLMs and LLM systems as reward-maximizing agents, and revealed theoretical relationships between our proposed methodology and the established notion of Pareto optimality.

Applying our framework to empirically exploring the practical relevance of non-reasoning LLMs and cascades, we found several interesting results. First, reasoning models offer superior accuracy-cost trade-offs on difficult mathematics questions as soon as the price of error exceeds \$0.01. Second, a single large LLM $\mathcal{M}_{\text{big}}$ typically outperforms a cascade $\mathcal{M}_{\text{big}} \to \mathcal{M}_{\text{big}}$ for prices of error as low as \$0.1.

Extrapolating these findings from mathematics to other domains, our results carry significant economic implications. We recommend that when automating meaningful human tasks with AI models, practitioners should typically use the most powerful available model, rather than attempt to minimize inference costs, since inference costs are likely dwarfed by the economic impact of AI errors. Fundamentally, this recommendation is based on the low costs per query of LLMs, which are increasingly negligible compared to human wages.

3.7 Limitations

First, our results on the MATH benchmark may be affected by data contamination, since the LLMs we evaluate may have been trained on similar questions. In addition, using Llama3.1 405B for correctness evaluation may artificially inflate this model's self-verification accuracy—however, we believe that this latter effect is limited since the prompt for correctness evaluation relies on access to the ground truth reference answer, whereas self-verification only incorporates the model's proposed (but possibly incorrect) answer.

Chapter 4

ERROR REDUCTION WITH HUMAN-IN-THE-LOOP

Zellinger, Michael J. and Matt Thomson (2025). "Fail Fast, or Ask: Mitigating the Deficiencies of Reasoning LLMs with Human-in-the-Loop Systems Engineering." In: *arXiv preprint*. arXiv: 2507.14406 [cs.AI]. URL: https://arxiv.org/abs/2507.14406.

Abstract: *State-of-the-art reasoning LLMs are powerful problem solvers, but they still occasionally make mistakes. However, adopting AI models in risk-sensitive domains often requires error rates near 0%. To address this gap, we propose collaboration between a reasoning model and a human expert who resolves queries the model cannot confidently answer. We find that quantifying the uncertainty of a reasoning model through the length of its reasoning trace yields an effective basis for deferral to a human, e.g., cutting the error rate of Qwen3 235B-A22B on difficult MATH problems from 3% to less than 1% when deferring 7.5% of queries. However, the high latency of reasoning models still makes them challenging to deploy on use cases with high query volume. To address this challenge, we explore fronting a reasoning model with a large non-reasoning model. We call this modified human-in-the-loop system "Fail Fast, or Ask", since the non-reasoning model may defer difficult queries to the human expert directly ("failing fast"), without incurring the reasoning model's higher latency. We show that this approach yields* $\approx$ 40% *latency reduction and* $\approx$ 50% *cost savings for DeepSeek R1 while maintaining 90+% area under the accuracy-rejection curve. However, we observe that latency savings are lower than expected because of latency drag—the phenomenon that processing easier queries with a non-reasoning model pushes the reasoning model's latency distribution towards longer latencies. Broadly, our results suggest that the deficiencies of state-of-the-art reasoning models—nontrivial error rates and high latency—can be substantially mitigated through black-box systems engineering, without requiring access to LLM internals.*

4.1 Introduction

Reasoning LLMs are trained using reinforcement learning to "think" until finding the correct answer to a query (DeepSeek AI, 2025). This approach has significantly

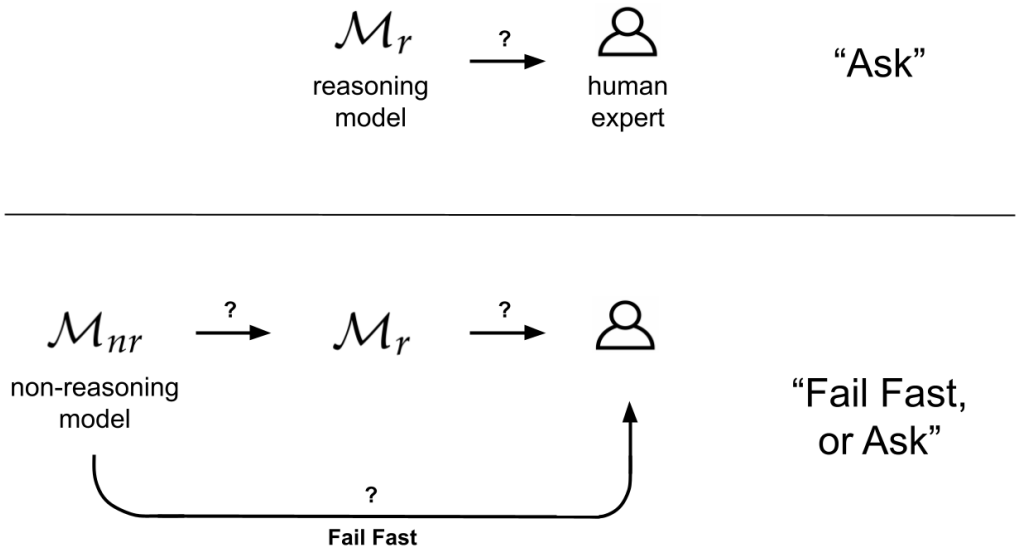


Figure 4.1: We explore two systems aimed at reducing deficiencies of reasoning models. Above: "Ask" aims to reduce the error rate of reasoning model $\mathcal{M}_r$ by deferring difficult queries to a human expert when $\mathcal{M}_r$ is uncertain. Bottom: "Fail Fast, or Ask" aims to offset the reasoning model's high latency by fronting it with a faster non-reasoning model $\mathcal{M}_{nr}$, which may directly defer queries to the human expert ("failing fast").

lowered error rates on complex tasks (OpenAI, 2024a; Alibaba AI, 2025), making "thinking" a dominant paradigm for state-of-the-art language models.

However, many risk-sensitive domains expect error rates near 0%—a high bar which even state-of-the-art reasoning models may fail to meet.

Moreover, reasoning models suffer from high latency (Zellinger and Thomson, 2025a), making these models unsuitable for tasks where answers must be delivered quickly. Query response times measured in seconds and even minutes can arise for difficult tasks, making standard user interaction patterns—which are based on rapid feedback—problematic (Kohavi and Longbotham, 2007). A similar challenge arises for batch workloads involving a large volume of queries. For example, with a per-query latency of 30 seconds—not unusual for reasoning models—crunching through a million database records takes around 1 year.

In this paper, we explore a human-in-the-loop systems approach ("Fail Fast, or Ask") for mitigating these deficiencies of reasoning models, with the goal of speeding their path towards practical deployment on real-world tasks.

Described simply, our "Fail Fast, or Ask" system applies distinct strategies for improving 1) the error rate and 2) the latency of a reasoning model $\mathcal{M}_r$.

To achieve error reduction, $\mathcal{M}_r$ defers difficult queries to a human expert $\mathcal{H}$. Similar to prior successes on achieving near-zero error rates on image classification (Geifman and El-Yaniv, 2017), this strategy lowers the error rate of $\mathcal{M}_r$ at the expense of abstaining from a nonzero fraction of queries.

Second, to speed up response times we send each query first to a faster, non-reasoning model $\mathcal{M}_{\text{nr}}$. This model may respond directly to the query, defer it to $\mathcal{M}_{\text{nr}}$, or send it to the human expert $\mathcal{H}$.

Our experiments yield the following conclusions:

1. Quantifying the uncertainty of reasoning models by the lengths of their thinking traces effectively improves accuracy for DeepSeek R1 and Qwen3 235B-A22B, but not OpenAI o3.

2. Deferring difficult queries to a human expert achieves +2% accuracy, pushing the accuracy of Qwen3 235B-A22B on the most challenging MATH questions from 97% to 99+%.

3. Our "Fail Fast, or Ask" system, which fronts a slow reasoning model with a faster non-reasoning model (Llama3.1 405B), maintains 90+% accuracy while responding to queries $\approx$ 40% faster.

4.2 Related Work

Quantifying the Uncertainty of Reasoning Models. Previous work suggests that longer reasoning traces correlate with lower accuracy (Ballon, Algaba, and Ginis, 2025), resulting from higher perceived problem difficulty (Shojaee et al., 2025; Lawsen, 2025).

Interestingly, this phenomenon highlights nuances of causation vs correlation. While longer reasoning traces *correlate* with higher error rates, deliberately using a greater number of reasoning tokens paradoxically *causes* improved performance (Jin et al., 2024; Muennighoff et al., 2025). Such apparent paradoxes are well-explained in the causal inference literature (Pearl, 2009) and are due to adverse selection effects. Specifically, when *forcing* a model to use a greater number of reasoning tokens, we

condition on all queries, whereas if we *observe* a model using a greater number of reasoning tokens, we condition on a particularly difficult set of queries.[1]

Selective Prediction with Large Language Models. Selective prediction (El-Yaniv and Wiener, 2010) gives a machine learning model the ability to abstain from answering difficult queries, making very low error rates possible. Using this approach, Geifman and El-Yaniv (2017)) have demonstrated near-zero error rates on top-5 ImageNet classification. In natural language processing (NLP), selective prediction has not been as popular, except for early contributions (Xin et al., 2021; Varshney, Mishra, and Baral, 2022; Ren et al., 2023a). Once NLP entered the LLM era, several authors have proposed uncertainty metrics for predicting LLM errors (Manakul, Liusie, and Gales, 2023; Azaria and Mitchell, 2023a; Chen, Zaharia, and Zou, 2023; Farquhar et al., 2024). These contributions typically evaluate the quality of their proposed uncertainty metrics by simulating a selective prediction task in which the LLM only answers queries for which its uncertainty is low. This evaluation results in a plot of conditional accuracy vs rejection rate, and the area under this curve (AUARC) quantifies performance across rejection rates.

Human-in-the-Loop AI Systems. Collaboration between AI systems and humans has been an active research topic in human-computer interaction (Amershi et al., 2014; Kamar, 2016; Shneiderman, 2020; Watkins et al., 2025), with heightened interest in application domains such as medicine. Some of this work has explored the complementary strengths of human experts and AI models (Dvijotham et al., 2023), while others treat human experts as an omniscient oracle—a safe fallback for imperfect AI models (Strong, Men, and Noble, 2025; Fanconi and Schaar, 2025). Our work takes this latter approach, although we view explicit modeling of human errors as a promising avenue for follow-up research.

4.3 Fail Fast, or Ask

We consider two human-in-the-loop systems, as illustrated in Figure 4.1. The first system ("Ask") consists of a large reasoning model $\mathcal{M}_r$ that defers difficult queries to a human expert $\mathcal{H}$. The second system ("Fail Fast, or Ask") additionally contains a large non-reasoning model $\mathcal{M}_{\text{nr}}$, which fields all queries; based on its confidence, $\mathcal{M}_{\text{nr}}$ either 1) responds to the query, 2) passes the query to the reasoning model $\mathcal{M}_{\text{r}}$, or 3) directly defers the query to the human expert ("failing fast").

[1] This scenario closely mirrors the common situation in hospitals where only the sickest patients are assigned the strongest medication. Naive statistical analysis then leads to the erroneous conclusion that the medication lowers patients' survival rate.

Predicting Reasoning Model Errors. Prior work has revealed that reasoning models emit more thinking tokens when they are uncertain about the answer to a query (Ballon, Algaba, and Ginis, 2025). Leveraging this insight, we predict reasoning model errors by simply thresholding the number of output tokens. For example, to reject 10% of queries (and defer them to the human expert), we set the output token threshold at the 90% quantile of its empirical distribution.

Predicting Non-Reasoning Model Errors. To predict the errors of $\mathcal{M}_{\text{nr}}$, we estimate its confidence using the P(True) strategy of Kadavath et al. (2022). Prior work shows that this methodology yields good performance for Llama3.1 405B (Zellinger and Thomson, 2025a).

On each query, the non-reasoning model chooses between three possible actions: 1) return the response to the query (`respond`), 2) pass the query to the reasoning model (`pass`), or 3) directly defer the query to the human expert (`fail fast`). Its action π_{nr} is based on simple thresholding of its P(True) confidence estimate p_{true}:

$$\pi_{nr} = \begin{cases} \texttt{fail fast} & \text{if } p_{\text{true}} \leq c_{\text{fail fast}}, \\ \texttt{pass} & \text{if } p_{\text{true}} > c_{\text{fail fast}} \text{ and } p_{\text{true}} \leq c_{\text{pass}}, \\ \texttt{respond} & \text{if } p_{\text{true}} > c_{\text{pass}}. \end{cases} \tag{4.1}$$

Configuration of the "Fail Fast, or Ask" System. To modulate utilization of the faster non-reasoning model $\mathcal{M}_{nr}$, we fix its utilization rate $u := \mathbb{P}(\pi_{nr} \neq \texttt{pass})$. This rate denotes the proportion of queries processed without involving the reasoning model $\mathcal{M}_r$; in other words, $1 - u$ is the rate at which $\mathcal{M}_{nr}$ passes queries to $\mathcal{M}_r$.

To set the rate $\mathbb{P}(\pi_{nr} = \texttt{fail fast})$, we start with a desired overall rejection rate $\mathbb{P}(\rightarrow \mathcal{H})$, where $\mathcal{H}$ represents the human expert. We enforce that the reasoning model's conditional rejection rate match the system's overall rejection rate:

$$\mathbb{P}(\mathcal{M}_r \rightarrow \mathcal{H} \mid \mathcal{M}_{nr} \rightarrow \mathcal{M}_r) = \mathbb{P}(\rightarrow \mathcal{H}).$$

This assumption sets the non-reasoning model's fail-fast rate at

$$\mathbb{P}(\pi_{nr} = \texttt{fail fast}) = \mathbb{P}(\rightarrow \mathcal{H}) - (1 - u)\, \mathbb{P}(\mathcal{M}_r \rightarrow \mathcal{H} \mid \mathcal{M}_{nr} \rightarrow \mathcal{M}_r).$$

This simple methodology yields an accuracy-rejection trade-off for each choice of the system's overall rejection rate $\mathbb{P}(\text{reject})$ and the non-reasoning model's utilization rate u, without requiring any training or optimization. Choosing $u > 0$ yields latency

and cost savings relative to the baseline "Ask" system consisting only of a reasoning model $\mathcal{M}_r$ and human expert (Figure 4.1), as Figure 4.5 demonstrates.

Human Expert. We assume that the human expert can instantly and perfectly answer each query. Thus, we treat deferral to the expert in the same manner as rejections of the query in selective prediction, and do not further model the human expert's error rate or the time he/she requires to resolve a query. However, it would be interesting to study these aspects in future work, as we note in Section 4.6.

Metrics. We measure system performance by the conditional accuracy $A(r)$ (or error $E(r) = 1 - A(r)$) for different "rejection rates" r of deferring queries to the human expert (El-Yaniv and Wiener, 2010). $A(r)$ is conditioned on the non-rejected queries; this ensures that accuracy does not improve automatically by raising the rejection rate. This approach yields *accuracy-rejection curves* charting $A(r)$ vs r. To summarize performance in a single number, we follow prior work and report the *area under the accuracy-rejection curve* (AUARC). We consider rejection rates from 0% to 20% and report AUARC as the mean conditional accuracy across these rejection rates.

4.4 Latency Drag

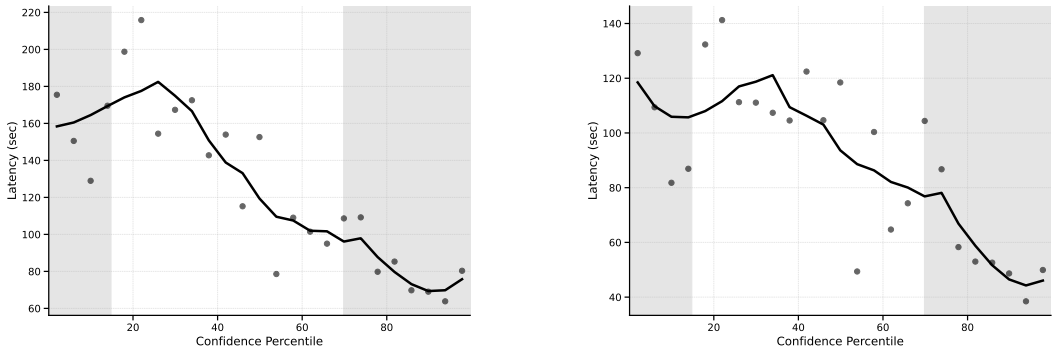
It may appear that deferring queries from the non-reasoning model $\mathcal{M}_{\text{nr}}$ to the reasoning model $\mathcal{M}_r$ at a rate $1 - u$ will yield a system latency of

$$\mathbb{E}[L] = u\,\mathbb{E}[L_{\text{nr}}] + (1-u)\,(\mathbb{E}[L_{\text{nr}}] + \mathbb{E}[L_{\text{r}}]), \tag{4.2}$$

where L_r is the latency of the reasoning model and L_{nr} is the latency of the non-reasoning model. Unfortunately, this expectation is mistaken, for a reason we term *latency drag*. Predicting errors by the reasoning model relies on the empirical correlation between high uncertainty and longer reasoning traces. Precisely this same correlation implies that conditioning on difficult queries (those passed by the non-reasoning model $\mathcal{M}_{\text{nr}}$) results in *longer latencies*.

Figure 4.2a shows the effect of latency drag for DeepSeek R1 and Qwen3 225B-A22. For lower confidence percentiles of Llama3.1 405B (x axis), the conditional latency of the reasoning models (y axis) rises. As a result, deferring queries from Llama3.1 405B to the reasoning model results in higher latencies than predicted by Equation (4.2).

More precisely, fronting the reasoning model $\mathcal{M}_r$ with a faster non-reasoning model alters the latency distribution of $\mathcal{M}_r$ in two ways, eliminating both low *and* high



(a) Qwen3 235B-A22B (b) DeepSeek R1

Figure 4.2: Fronting the reasoning model $\mathcal{M}_r$ with a non-reasoning model $\mathcal{M}_{nr}$ increases the reasoning model's average latency (white area), leading to lower latency savings than expected. Fundamentally, this *latency drag* results from the negative correlation between the reasoning model's latency (y axis) and the non-reasoning model's confidence (x axis).

latencies: the former by never passing easy queries, and the latter by "failing fast" on the most difficult queries. The gray shaded areas in Figure 4.2 illustrates these two competing effects. In practice, latency drag from eliminating easy queries outweighs latency reduction from "failing fast."

4.5 Selective Prediction Performance

We evaluate the selective prediction performance of both the "Ask" system $\mathcal{M}_r \to \mathcal{H}$ and the "Fail Fast, or Ask" system $[\mathcal{M}_{nr} \to \mathcal{M}_r] \to \mathcal{H}$, for different degrees of utilizing the non-reasoning model.

To address the utility of our methodology in a complex problem-solving domain where reasoning models excel, we evaluate on 500 questions with maximum difficulty (5/5) from the MATH benchmark (Hendrycks et al., 2021b). We use the same data as Zellinger and Thomson (2025a), which is filtered for questions with numeric answers.

We consider three state-of-the-art large reasoning models: Qwen3 225B-A22 (Alibaba AI, 2025), DeepSeek R1 (DeepSeek AI, 2025), and OpenAI o3. As the non-reasoning model, we use Llama3.1 405B (Meta AI, 2024) throughout, since prior work suggests it effectively quantifies its own uncertainty on MATH (Zellinger and Thomson, 2025a). Table 4.1 summarizes the baseline performance of these LLMs.

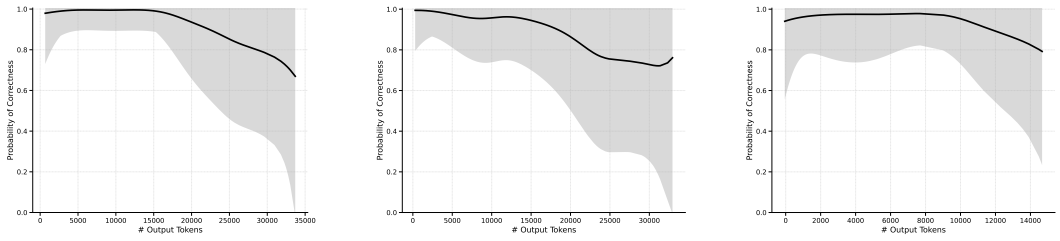
Table 4.1: Baseline performance metrics for reasoning ($\mathcal{M}_r$) and non-reasoning ($\mathcal{M}_{nr}$) models for difficult MATH questions. For each metric, we report the average value per query.

Model	Role	Error Rate (%)	Latency (sec)	Cost ($)	Output Tokens (#)
qwen3-235b-a22b	$\mathcal{M}_r$	2.8	125.9	9.5×10^{-3}	10.8K
deepseek-r1-0528	$\mathcal{M}_r$	5.8	89.7	7.9×10^{-2}	9.8K
o3-2025-04-16	$\mathcal{M}_r$	3.8	67.6	1.9×10^{-2}	2.2K
llama-v3p1-405b-instruct	$\mathcal{M}_{nr}$	30.6	12.4	3.6×10^{-3}	978

How well can we quantify the uncertainty of reasoning models?

Figure 4.3 shows the results of using the length of reasoning models' thinking traces to predict their correctness on answering queries. Each plot shows a local linear regression computing the conditional expectation of correctness for a specified number of output tokens. Each curve shows decreased accuracy when the number of output tokens is large.

However, we observe some variability between the models. Although the shape of each curve is similar—initially flat, then steeply declining—this pattern is attenuated for OpenAI o3, which displays a longer flat region and smaller decline in accuracy.



(a) Qwen3 235B-A22B (b) DeepSeek R1 (c) OpenAI o3

Figure 4.3: Local linear regression ($\pm 1\sigma$) of the reasoning models' correctness vs the number of output tokens shows that correctness decreases when reasoning traces are long.

How low does selective prediction push the error rates of reasoning models?

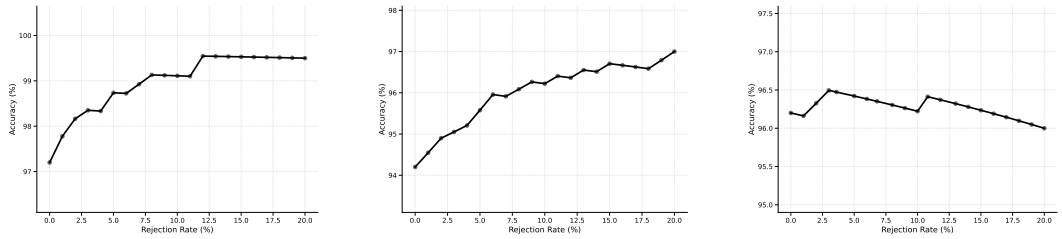
Since the length of reasoning models' thinking traces correlates with decreased accuracy, queries leading to long reasoning traces carry a high risk of error. How much can we raise a model's accuracy by deferring such "risky" queries to a human expert? Figure 4.4 shows curves of the conditional accuracy

$$\text{Accuracy} = \mathbb{E}[\text{Correct} \mid \#\text{ Output Tokens} \leq T] \tag{4.3}$$

for different thresholds T of the number of output tokens. We consider thresholds yielding rejection rates from 0% to 20% (x axis).

The figure reveals that both Qwen3 235B-A22B and DeepSeek R1 yield increased accuracy when avoiding "risky" queries leading to long reasoning traces. For the Qwen3 model, deferring 7.5% of queries reduces the error rate to 1% from 3%—a 2% reduction. For DeepSeek R1, accuracy gradually improves from 94% to 97% when deferring up to 20% of queries, although the majority of the gains (+2% accuracy) are already attained for a lower rejection rate of 7.5%.

By contrast, OpenAI o3 does not yield increased accuracy from avoiding "risky" queries with long reasoning traces. We speculate that the diminished efficacy of thinking duration as a proxy for uncertainty stems from OpenAI's proprietary optimization of the thinking process, compared to the simple reinforcement learning recipes described in the technical reports for the open-source models (DeepSeek AI, 2025; Alibaba AI, 2025). In the absence of other techniques for quantifying the uncertainty of o3, this finding places o3 at a disadvantage relative to other reasoning models.



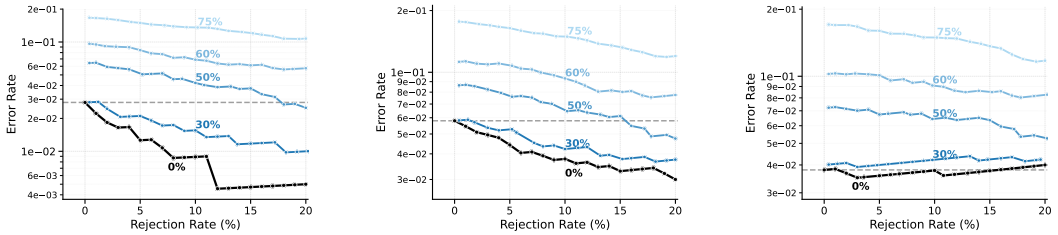
(a) Qwen3 235B-A22B (b) DeepSeek R1 (c) OpenAI o3

Figure 4.4: Abstaining from answering difficult queries based on the length of the reasoning trace yields 99+% accuracy for Qwen3 235B-A22B and 97% accuracy for DeepSeek R1, but yields no performance gains for OpenAI o3.

What latency and cost savings does "Failing Fast" yield?

Deferring risky queries from the reasoning model $\mathcal{M}_r$ to a human expert reduces AI errors. However, it does not address the high latency of reasoning models, which is problematic for interactive use cases or batch workloads with a large volume of queries. Figure 4.5 shows the error-rejection trade-offs when fronting the reasoning model $\mathcal{M}_r$ by a faster non-reasoning model $\mathcal{M}_{nr}$, as described in Section 4.3.

The figure shows that when using Qwen3 235B-A22B or DeepSeek R1 as the reasoning model, processing 30% to 75% of queries with Llama3.1 405B yields exponentially decreasing error for rejection rates from 0% to 20%. However, error decreases faster when using only the reasoning model (indicated as 0% in the figure, cf. Figure 4.4). Notably, 50% utilization of Llama3.1 405B with 15-20% deferral



(a) Qwen3 235B-A22B (b) DeepSeek R1 (c) OpenAI o3

Figure 4.5: Different degrees of non-reasoning model utilization (% values annotated in color) yield parallel error curves that decrease exponentially with increasing rejection rate (y axis is log-scale). 0% indicates using only the reasoning model. The dashed gray line indicates the reasoning model's baseline error rate.

Table 4.2: Human-in-the-loop reasoning model fronted by non-reasoning model maintains high selective prediction performance (90%+ AUARC ↑) while saving up to 38% latency and 46% cost. %Δ denotes performance drop from base AUARC as percentages. ΔL(%) and ΔC(%) denote percentage drops in latency and cost.

Model	**Base**	**50% Non-Reasoning**				**60% Non-Reasoning**				**75% Non-Reasoning**			
	AUARC	AUARC	Δ	ΔL(%)	ΔC(%)	AUARC	Δ	ΔL(%)	ΔC(%)	AUARC	Δ	ΔL(%)	ΔC(%)
deepseek-r1-0528	0.96	0.93	**-2.8**	-26.2	**-35.7**	0.91	**-5.4**	-36.5	**-45.8**	0.85	**-11.3**	-54.8	**-64.1**
qwen3-235b-a22b	**0.99**	**0.96**	-3.4	**-28.4**	-2.1	**0.93**	-6.2	**-37.8**	-11.6	**0.86**	-12.6	**-55.7**	-29.7
o3-2025-04-16	0.96	0.94	**-2.8**	-13.4	-14.8	0.91	-5.5	-23.2	-24.9	0.85	-11.4	-47.5	-44.4

suffices to push the systems' error rates below the reasoning models' baseline error rates.

Table 4.2 quantifies the latency and cost savings resulting from 50%, 60%, and 75% utilization of the non-reasoning model. To measure performance holistically across rejection rates, we report the area under the accuracy-rejection curve (AUARC), that is, the mean accuracy across rejection rates from 0% to 20%. The table shows that 50% utilization of Llama3.1 405B incurs a 3% drop in AUARC, but cuts latency by ≈ 27% (Qwen3 235B-A22B and DeepSeek R1), and cost by ≈ 36% (DeepSeek R1). Raising utilization to 60% yields higher latency reductions of ≈ 40% while still maintaining AUARC above 90%—significantly outperforming Llama3.1 405B's baseline accuracy of ≈ 69%.

Finally, 75% utilization of Llama3.1 405B reduces AUARC to the mid-80% range but yields correspondingly larger reductions in latency (≈ 55% for Qwen3 and DeepSeek R1) and cost (64% for DeepSeek R1).

4.6 Discussion

To mitigate the practical deficiencies of a reasoning LLMs—imperfect accuracy and high latency—we have embedded a reasoning model $\mathcal{M}_r$ into a larger system including a fast non-reasoning model $\mathcal{M}_{nr}$ and a human expert $\mathcal{H}$. As figures 4.5a and 4.5b show, this system is capable of beating the reasoning model's accuracy while cutting the latency by around 30%.

However, our methodology treats the human expert as an omniscient oracle. One way to model $\mathcal{H}$ is *learning to defer* (Madras, Pitassi, and Zemel, 2018). However, this approach adapts the AI model to the human, which seems impractical for API-based usage of large reasoning LLMs. Instead of attempting to adapt a trillion-parameter neural network that cost millions of dollars to train, it appears more sensible to adapt the human expert to the LLM.

For example, suppose the human expert $\mathcal{H}$ makes mistakes at a rate $\mathbb{E}[\mathbb{1}^{\mathcal{H}}_{\text{error}}]$. It is clear that we should select $\mathcal{H}$ to minimize this error rate. However, there is an important nuance. It is critical to minimize the error rate over the *difficult queries*,

$$\mathbb{E}_{q \sim p_{\text{difficult}}}[\mathbb{1}^{\mathcal{H}}_{\text{error}}(q)], \tag{4.4}$$

where $p_{\text{difficult}}(\cdot)$ samples queries for which the reasoning model has a high error rate. In other words, we wish to avoid an "accuracy drag" whereby a correlation between the AI model's and the human's perceptions of difficulty raises the error rate of $\mathcal{H}$ conditioned on the deferral decision $\mathcal{M}_r \to \mathcal{H}$.

This observation has practical implications. For example, suppose we wish to construct a human-in-the-loop system for solving challenging mathematics problems. Among the pool of possible mathematics experts, many of the most able mathematicians likely underperform the LLM on mechanical or otherwise unstimulating questions, simply because they find such questions boring. However, they would likely outperform on "interesting" questions whose solutions require inspiration and imagination—the same problems on which AI models are most likely to make mistakes (we speculate). Hence, a screening exam for identifying suitable mathematics experts should only include "interesting" questions whose solutions require ingenuity; adding any easier problems would be counterproductive.

4.7 Conclusion

This paper explored reducing the error rate and latency of a reasoning model by embedding it within a larger system that includes a human expert.

First, our experiments show that deferring difficult queries to a human expert can yield meaningful error reductions. Leveraging selective prediction based on the length of the thinking trace, we lowered the conditional error rate of Qwen3 235B-A22B and DeepSeek R1 by around 2% on difficult MATH questions—from 2.8% to 0.5% and 5.8% to 3%, respectively. Second, we found that fronting a reasoning model with a large non-reasoning model yields substantial ($\tilde{3}0\%$) latency reductions while outperforming the reasoning model's baseline accuracy. Unfortunately, these latency savings are lower than expected because of *latency drag*—the phenomenon that conditioning on difficult queries shifts the reasoning model towards longer latencies than usual.

Overall, our work offers a novel perspective for adapting reasoning LLMs to the demands of practical use cases. In future work, we are interested in circumventing latency drag by quantifying the uncertainty of a reasoning model in a manner uncorrelated with its latency. In addition, we aim to explicitly model the capabilities of human experts.

Chapter 5

PROOF-OF-CONCEPT SYSTEM FOR SYNTHETIC DATA GENERATION

Zellinger, Michael J. and Peter Bühlmann (2025). "Natural Language-Based Synthetic Data Generation for Cluster Analysis." In: *Journal of Classification*. DOI: 10.1007/s00357-025-09501-w. URL: https://doi.org/10.1007/s00357-025-09501-w.

Abstract: *Cluster analysis relies on effective benchmarks for evaluating and comparing different algorithms. Simulation studies on synthetic data are popular because important features of the data sets, such as the overlap between clusters, or the variation in cluster shapes, can be effectively varied. Unfortunately, creating evaluation scenarios is often laborious, as practitioners must translate higher-level scenario descriptions like "clusters with very different shapes" into lower-level geometric parameters such as cluster centers, covariance matrices, etc. To make benchmarks more convenient and informative, we propose synthetic data generation based on direct specification of high-level scenarios, either through verbal descriptions or high-level geometric parameters. Our open-source Python package* `repliclust` *implements this workflow, making it easy to set up interpretable and reproducible benchmarks for cluster analysis. A demo of data generation from verbal inputs is available at* `demo.repliclust.org`*.*

5.1 Introduction

The goal of clustering is to separate data points into groups such that points within a group a more similar to each other than to those outside the group (Cormack, 1971). In practice, it is often not clear what constitutes a cluster (Hennig, 2015). As a result, many practitioners evaluate cluster analysis algorithms on synthetic data (Milligan, 1980; Milligan, Soon, and Sokol, 1983; Tibshirani, Walther, and Hastie, 2001; Steinley and Brusco, 2008; Steinley and Brusco, 2011; Van Mechelen et al., 2023).

Synthetic data is valuable for two reasons. First, it clearly stipulates which data points belong to which cluster, allowing objective evaluation. Second, it allows independently manipulating different aspects of the data (such as the overlap between

clusters or the variability of cluster shapes), which is critical for drawing scientific conclusions about the relative merits of different cluster analysis techniques (Milligan, 1996).

Unfortunately, setting up benchmarks with synthetic data can be laborious. The process typically involves creating data sets for a number of different scenarios. For example, on benchmarks with convex clusters drawn from probabilistic mixture models, the scenarios may involve "clusters of very different shapes and sizes," "highly overlapping oblong clusters," "high-dimensional spherical clusters," etc. (Tibshirani, Walther, and Hastie, 2001).

Existing data generators do not cater directly to such high-level scenarios. Instead, the user must carefully tune simulation parameters to arrive at the desired scenarios (Steinley and Henson, 2005; Schubert and Zimek, 2019; Iglesias et al., 2019). While some generators make it easy to control the overlaps between clusters, such high-level control typically does not extend to other aspects like the desired diversity of cluster shapes and sizes (Qiu and Joe, 2006; Shand et al., 2019).

In this paper, we explore generating synthetic data directly from high level descriptions. Our Python package, `repliclust`, accomplishes this goal by summarizing the overall geometry of probabilistic mixture models with a few high-level parameters. We use a large language model to map a user's verbal description of a scenario onto these parameters. Although our approach is based on ellipsoidal clusters, we have implemented two post-processing functions for generating more irregular cluster shapes. The first makes clusters non-convex by passing them through a randomly initialized neural network. The second makes a p-dimensional data set *directional* by wrapping it around the $(p+1)$-dimensional sphere through an inverse stereographic projection.

5.2 Generating Data from High-Level Archetypes

Our data generator `repliclust` is based on data set archetypes. A *data set archetype* is a high-level description of the overall geometry of a data set with clusters. For example, the class of all data sets with "three oblong and slightly overlapping clusters in two dimensions with some class imbalance" is a data set archetype.

We implement archetype-based generation by summarizing the overall geometry of probabilistic mixture models in terms of a few high-level parameters, as listed in Table 5.1. To create an archetype, users can either specify these parameters directly or verbally describe the archetype in English. To map an English description to

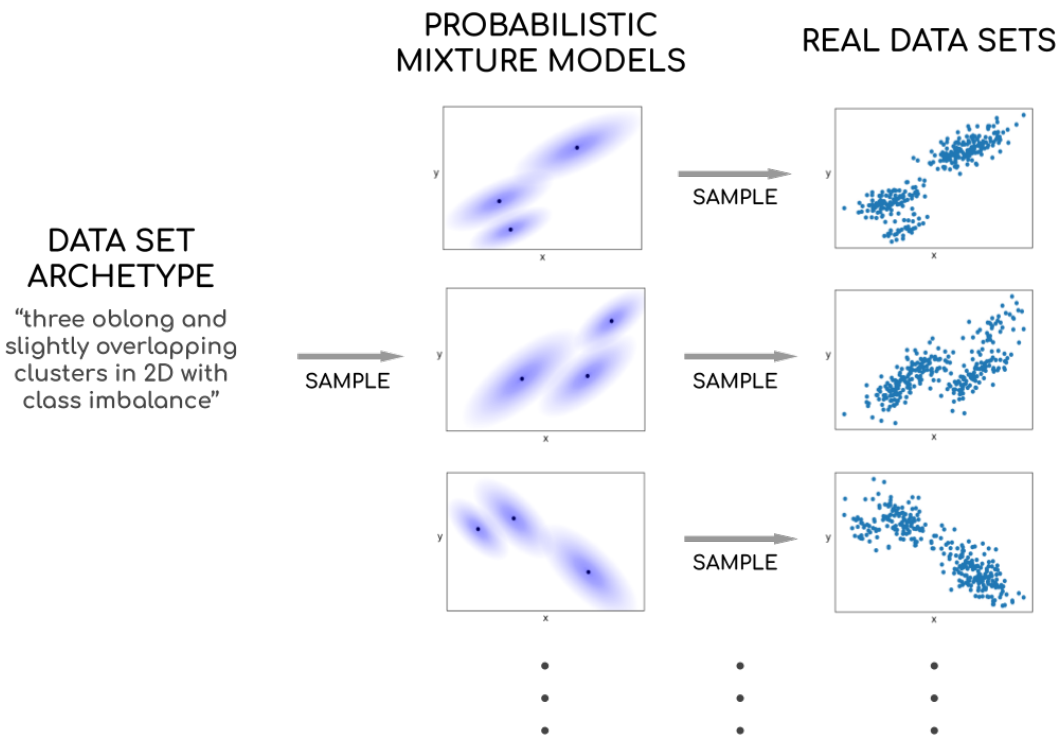


Figure 5.1: Illustration of synthetic data generation with data set archetypes. Left: the user specifies the desired archetype. The user can verbally describe the archetype in English or directly specify a few high-level geometric parameters. Middle: the archetype provides a random sampler for probabilistic mixture models with the desired geometric characteristics. Right: drawing i.i.d. samples from each mixture model yields synthetic data sets.

a precise parameter setting, we use few-shot prompting of a large language model (Brown et al., 2020a; OpenAI, 2024b).

Table 5.1: Summary of high-level geometric parameters defining an `Archetype` in `repliclust`.

Parameter(s)	**Purpose**
`n_clusters` / `dim` / `n_samples`	select number of clusters / dimensions / data points
`aspect_ref` / `aspect_maxmin`	determine how elongated vs spherical clusters are / how much this varies between clusters
`radius_maxmin`	determine the variation in cluster volumes
`max_overlap` / `min_overlap`	set maximum / minimum overlaps between clusters
`imbalance_ratio`	make some clusters have more data points than others
`distributions` / `distribution_proportions`	select probability distributions appearing in each data set / how many clusters have each distribution[1]

Most of the high-level geometric parameters describing an archetype are based on

what we call "max-min sampling." In this approach, the user controls a geometric attribute by specifying a *reference value* and *max-min ratio*. In addition, a constraint ensures that the reference value and is indeed typical for *every* data set. For example, the aspect ratio of a cluster measures how elongated it is. The reference value `aspect_ref` sets the typical aspect ratio among all clusters in a data set, while `aspect_maxmin` sets the ratio of the highest to the lowest aspect ratio. To make sure that `aspect_ref` is indeed the typical aspect ratio, max-min sampling enforces the location constraint $(\prod_{j=1}^{k} \alpha_j)^{\frac{1}{k}} =$ `aspect_ref`. Appendix A gives more details on how we manage different geometric attributes using max-min sampling.

Once an archetype has been defined, sampling concrete data sets proceeds in two steps. First, the algorithm samples a new probabilistic mixture model whose geometric structure matches the archetype. Second, we draw i.i.d. samples from this mixture model to generate a data set. Figure 5.1 illustrates this flow.

To accommodate use cases in which variation of an archetype's hyperparameters (`n_clusters`, `dim`, `n_samples`) is desired, we have implemented a function `Archetype.sample_hyperparams` for generating a list of archetypes with hyperparameters sampled from Poisson distributions centered on the original values (subject to rejection sampling based on user-specified minimum and maximum values).

5.3 Sampling Probabilistic Mixture Models

Generating a synthetic dataset with `repliclust` starts with sampling a probabilistic mixture model that matches the desired archetype.

Sampling a mixture model proceeds through the following steps:

1. Draw random cluster covariance matrices based on the archetype.

2. Randomly initialize the cluster centers. Adjust their placement using stochastic gradient descent to meet desired constraints on the overlaps between clusters.

3. Sample the number n_j of data points per cluster, based on the extent of class imbalance specified by the archetype.

4. Construct a data set X and cluster labels y by sampling n_j data points i.i.d. from the mixture component describing cluster j. Return $(X, y, \mathcal{A})$, where $\mathcal{A}$ is the archetype.

5. Optionally, make cluster shapes non-convex by either

    a) passing X through a randomly initialized neural network (`repliclust.distort`)

    b) wrapping $X \in \mathbb{R}^{n \times p}$ around the $(p+1)$-dimensional sphere to create directional data (`repliclust.wrap_around_sphere`).

In the following sections, we give more details on each step involved in data generation.

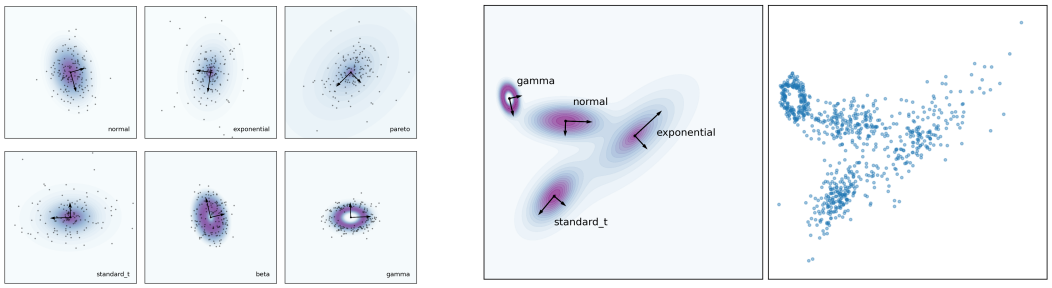
Defining Clusters and Mixture Models

In the first four steps of data generation, `repliclust` models clusters as ellipsoidal probability distributions characterized by a central point. Specifically, each cluster C is defined by a cluster center $\mathbf{x}_C \in \mathbb{R}^p$, orthogonal principal axes $\hat{\boldsymbol{u}}_C^{(1)}, \hat{\boldsymbol{u}}_C^{(2)}, ..., \hat{\boldsymbol{u}}_C^{(p)}$ pointing in arbitrary directions, the lengths $\sigma_C^{(1)}, \sigma_C^{(2)}, ..., \sigma_C^{(p)}$ of the principal axes, and a univariate probability distribution $f_C(\cdot)$.

To generate an i.i.d. sample from a cluster, we 1) sample the direction $\hat{\mathbf{x}}$ from the ellipsoid defined by the cluster's principal axes, and 2) sample the length $||\mathbf{x}||_2$ according to the cluster's univariate distribution f_C (which can be one of many supported distributions including normal, lognormal, exponential, Student's t, gamma, chi-square, Weibull, Gumbel, F, Pareto, beta, and uniform). To make the spread of each cluster depend only on the lengths of the principal axes, we normalize each univariate distribution so that the 68.2% quantiles of its absolute value is unity. For example, if the univariate distribution is exponential with rate λ, we would actually sample from a re-scaled random variable $\text{Exp}(\lambda)/q_{0.682}$, where the quantile $q_{0.682}$ satisfies $\mathbb{P}(|\text{Exp}(\lambda)| \leq q_{0.682}) = 0.682$. This rescaling puts all distributions on the same scale as the multivariate normal distribution, which natively satisfies $\mathbb{P}(|\mathcal{N}(0,1)| \leq 1) \approx 0.682$.

Figure 5.2(a) visualizes clusters with different base distributions. Note that using heavy-tailed distributions can lead to far outliers (not shown). By contrast, univariate distributions with bounded support give rise to clusters with crisply defined boundaries. Stretches where a probability density function vanishes give rise to concentric holes.

A mixture model (`repliclust.MixtureModel`) represents several clusters. Figure 5.2(b) shows a two-dimensional mixture model with four clusters, and a data set sampled from it. Note that our mixture models do not model class probabilities. Instead, the number of data points per cluster are drawn using max-min sampling,



(a) (b)

Figure 5.2: Individual clusters (a) and a probabilistic mixture model (b) in `repliclust`. Black arrows show each cluster's principal axes. The scatter plot on the right in (b) shows a data set sampled from the mixture model. In this example, all clusters are natively 2D.

based on the archetype's imbalance ratio (desired ratio between the number of data points in the most and least populous clusters). Appendix B lists the formal attributes of a mixture model.

Managing Cluster Overlaps

Managing the degree of overlaps between clusters is one of the most important tasks of a cluster generator. In `repliclust`, we quantify pairwise overlap between two clusters as the irreducible error rate when classifying a new data point as belonging to one of the two clusters (assuming equal class probabilities). On the level of entire datasets, the user controls overlap by specifying the *maximum* and *minimum* pairwise overlap.

The maximum overlap `max_overlap` states that no pair of clusters may overlap more than a certain amount. By contrast, the minimum overlap `min_overlap` prevents isolated clusters by enforcing that each cluster has *some* neighbor with which it overlaps *at least* `min_overlap`. We construct a loss function that enforces the minimum and maximum overlap conditions and optimize it using stochastic gradient descent (SGD).

In this section, we first describe our notion of pairwise overlap. Second, we explain the loss function we use to enforce the maximum and minimum overlap for an entire dataset.

Measuring Pairwise Overlap

Viewing clusters as probability distributions, we formally define the overlap between two clusters as *twice the minimax error rate when classifying new data points using a linear decision boundary between the two clusters*. Figure 5.3 illustrates this definition. To explain what we mean by "minimax" in this context, observe that any linear classifier $\hat{y} : \mathbb{R}^p \mapsto \{1, 2\}$ depends on an axis $\boldsymbol{a} \in \mathbb{R}^p$ and threshold $c \in \mathbb{R}$ such that

$$\hat{y}(\mathbf{x}) = \begin{cases} 1, & \text{if } \boldsymbol{a}^\top \mathbf{x} \leq c \\ 2, & \text{if } \boldsymbol{a}^\top \mathbf{x} > c. \end{cases} \tag{5.1}$$

By definition, the *minimax* classifier $\hat{y}^*$ minimizes the worst-case loss. In symbols,

$$\max_y \ \mathbb{P}(y \neq \hat{y}^*(\mathbf{x})|y) = \min_{\hat{y}} \max_y \ \mathbb{P}(y \neq \hat{y}(\mathbf{x})|y), \tag{5.2}$$

where $y \in \{1, 2\}$ is the true cluster label corresponding to a new data point $\mathbf{x} \in \mathbb{R}^p$. The outer minimum on the right hand side ranges over all linear classifiers $\hat{y}$, including $\hat{y}^*$. Rewriting (5.2) in terms of the classification axes $\boldsymbol{a}$ and thresholds c yields

$$\boldsymbol{a}^*, \ c^* = \underset{\boldsymbol{a} \in \mathbb{R}^p, \ c \in \mathbb{R}}{\arg\min} \ \max\{ \ \mathbb{P}(\boldsymbol{a}^\top \mathbf{x} > c \mid y = 1), \ \mathbb{P}(\boldsymbol{a}^\top \mathbf{x} \leq c \mid y = 2) \ \}. \tag{5.3}$$

It is not hard to see that the minimax condition requires the cluster-specific error rates $\mathbb{P}(\boldsymbol{a}^{*\top} \mathbf{x} > c^* \mid y = 1)$ and $\mathbb{P}(\boldsymbol{a}^{*\top} \mathbf{x} \leq c^* \mid y = 2)$ to be equal. Consequently, the cluster overlap α becomes

$$\alpha = 2 \, \mathbb{P}(\boldsymbol{a}^{*\top} \mathbf{x} > c^* \mid y = 1). \tag{5.4}$$

Geometrically, our definition means that two clusters overlap at level α if their marginal distributions along the minimax classification axis $\boldsymbol{a}^*$ intersect at the $1 - \alpha/2$ and $\alpha/2$ quantiles. The left panel of Figure 5.3 highlights the probability mass bounded by these quantiles in gray.

Note that our formulation of *minimax* classification error explicitly does not take into account class probabilities for the clusters. Equation (5.2) depends only on the class-conditional probabilities. Our reasoning is that the underlying reality of each cluster depends on its class-conditional probability distribution, not the class probability.

Steinley and Henson, 2005 quantify cluster overlap by computing the full distributional overlap in p dimensions. We prefer our one-dimensional notion in terms of

minimax classification error, since high-dimensional Euclidean geometry exhibits a number of counterintuitive phenomena. Specifically, as $p \to \infty$, the majority of a sphere's volume becomes concentrated in a thin shell from its surface, and so p-dimensional overlap goes to zero unless the marginal overlaps in each dimension approach 100% (Blum, Hopcroft, and Kannan, 2020; Steinley and Henson, 2005).

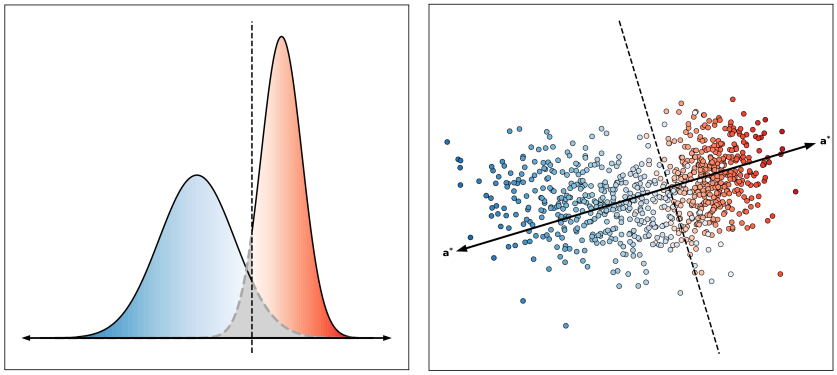


Figure 5.3: Cluster overlap based on the misclassification rate of the best linear classifier, in 1D (left) and 2D (right). The black dashed lines show the decision boundaries corresponding to minimax classification rules between the blue and red clusters, and the gray shaded areas represent classification errors. Cluster overlap α is the total probability mass of the gray areas. Here, $\alpha = 14.7\%$ for both the left and right panels.

Computing our cluster overlap α requires finding the minimax classification axis $\boldsymbol{a}^*$. Anderson and Bahadur, 1962 describe an algorithm for computing $\boldsymbol{a}^*$ exactly, in the case of multivariate normal distributions. Unfortunately, their method requires computing the matrix inverse $(t\boldsymbol{\Sigma}_1 + (1-t)\boldsymbol{\Sigma}_2)^{-1}$ for $O(\log(1/\epsilon)$ values of t, where $\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2$ are the clusters' covariance matrices and ϵ is a numerical tolerance. To avoid these matrix inversions, we propose an approximation of the minimax classification axis based on linear discriminant analysis (LDA).

For a pair of multivariate normal clusters with means $\boldsymbol{\mu}_1 \neq \boldsymbol{\mu}_2$ and the same covariance matrix $\boldsymbol{\Sigma}$, the axis $\boldsymbol{a}_{\text{LDA}} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)$ minimizes the overall misclassification rate, assuming equal class probabilities (see Hastie, Tibshirani, and Friedman, 2009). Unfortunately, the result does not hold for unequal covariance matrices $\boldsymbol{\Sigma}_1 \neq \boldsymbol{\Sigma}_2$. Thus, we propose the approximation $\boldsymbol{a}_{\text{LDA}} = (\frac{\boldsymbol{\Sigma}_1+\boldsymbol{\Sigma}_2}{2})^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)$. Figure 5.4 verifies that this LDA-based approximation closely matches the exact overlap, as compared with a simpler "center-to-center" (C2C) approximation that uses the difference between the cluster centers as the classification axis (i.e., $\mathbf{a}_{C2C} = \boldsymbol{\mu}_2 - \boldsymbol{\mu}_1$). In the figure, each data point corresponds to a pair of multivariate normal clusters with

the pairwise overlap shown. The full simulation is based on 900 pairs generated from a variety of data set archetypes with different cluster shapes and numbers of dimensions (see code repository for implementation details).

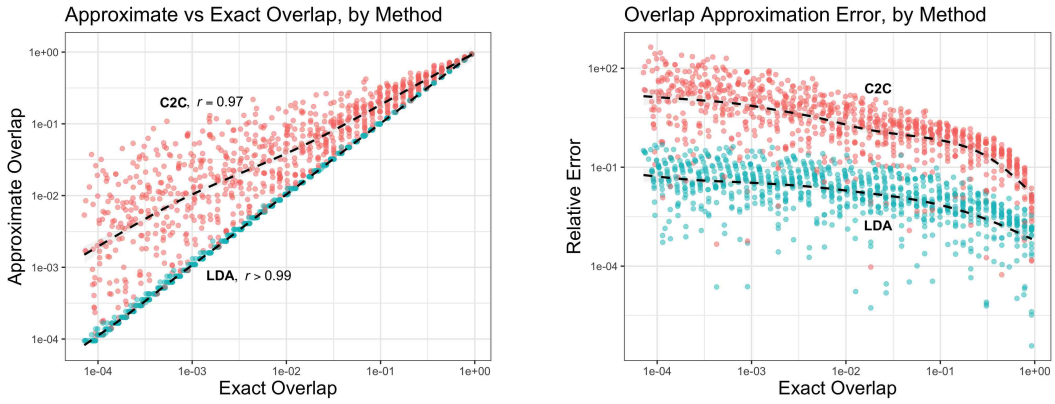


Figure 5.4: Quality of approximating cluster overlap using our LDA and the simpler center-to-center (C2C) approximations. Both approaches show strong correlations with exact cluster overlap, achieving Pearson correlations r close to 1 (left). However, the C2C method incurs significant relative error, while the LDA approximation typically comes within 10% of the exact overlap (right). The dashed lines indicate estimated conditional means.

Theorem 6 gives a formula for computing the LDA-approximate cluster overlap. The proof is in Appendix C.

Theorem 6 (LDA-Based Cluster Overlap)**.** *For two multivariate normal clusters with means $\boldsymbol{\mu}_1 \neq \boldsymbol{\mu}_2$ and covariance matrices $\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2$, the approximate cluster overlap α_{LDA} based on the linear separator $\boldsymbol{a}_{LDA} = (\frac{\boldsymbol{\Sigma}_1+\boldsymbol{\Sigma}_2}{2})^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)$ is*

$$\alpha_{LDA} = 2\Big(1 - \Phi\Big(\frac{\boldsymbol{a}_{LDA}^\top(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)}{\sqrt{\boldsymbol{a}_{LDA}^\top \boldsymbol{\Sigma}_1 \boldsymbol{a}_{LDA}} + \sqrt{\boldsymbol{a}_{LDA}^\top \boldsymbol{\Sigma}_2 \boldsymbol{a}_{LDA}}}\Big)\Big), \tag{5.5}$$

where $\Phi(z)$ is the cumulative distribution function of the standard normal distribution. Moreover, if $\boldsymbol{\Sigma}_1 = \lambda \boldsymbol{\Sigma}_2$ for some λ then α_{LDA} equals the exact cluster overlap α.

Theorem 6 shows that cluster overlap depends on quantiles q of the form

$$q(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2; \boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2; \boldsymbol{a}) = \frac{\boldsymbol{a}^\top(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)}{\sqrt{\boldsymbol{a}^\top \boldsymbol{\Sigma}_1 \boldsymbol{a}} + \sqrt{\boldsymbol{a}^\top \boldsymbol{\Sigma}_2 \boldsymbol{a}}}, \tag{5.6}$$

where $\boldsymbol{a}$ is a classification axis. Since these quantiles are inversely related to cluster overlap, they quantify cluster *separation*.

When generating synthetic data, `repliclust` uses Theorem 6 even when cluster distributions are non-normal. Figure .15 in Appendix D suggests that controlling overlap between non-normal distributions works well, except for distributions with bounded support (such as beta).

Adjusting Cluster Centers to Meet Overlap Constraints

To enforce maximum and minimum overlaps for a mixture model, we optimize a loss function using stochastic gradient descent (SGD). Given mixture model parameters $\boldsymbol{\theta}$ (including the cluster centers, principal axes, and principal axis lengths), the *overlap loss* is

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{k} \sum_{i=1}^{k} \ell_i, \tag{5.7}$$

where the loss on the i-th cluster is

$$\ell_i = p_\lambda\big((\min_{j \neq i} q_{ij} - q_{\max})^+\big) + \sum_{j \neq i} p_\lambda\big((q_{\min} - q_{ij})^+\big). \tag{5.8}$$

Here, $q_{\min}$, $q_{\max}$, and q_{ij} measure cluster separation as expressed in Equation (5.6): $q_{\min}$ is the minimum allowed separation (corresponding to overlap $\alpha =$ `max_overlap`); $q_{\max}$ is the maximum allowed separation (corresponding to $\alpha =$ `min_overlap`); and q_{ij} is the separation between the i-th and j-th clusters. The penalty function p_λ is the polynomial $p_\lambda(x) := \lambda x + (1 - \lambda)x^2$, where λ is a tuning parameter. Finally, $x^+ := \max(x, 0)$ is a filter that passes on only positive inputs (corresponding to a violation of user-specified constraints).

Intuition behind the Overlap Loss

By design, the loss (5.8) vanishes when the cluster centers, principal axes, and principal axis lengths satisfy the user-specified overlap constraints. The first term penalizes violation of the minimum overlap condition. Indeed, if cluster i is too far away from the other clusters, the separation $\min_{j \neq i} q_{ij}$ between cluster i and its closest neighbor exceeds the maximum allowed separation $q_{\max}$. A penalty of the excess $(\min_{j \neq i} q_{ij} - q_{\max})^+$ yields the first term in (5.8). The second term measures violation of the maximum overlap condition. If the separation q_{ij} between clusters i and j falls short of the smallest allowed separation $q_{\min}$, the shortfall $(q_{\min} - q_{ij})^+$ incurs a penalty that serves to push these clusters apart.

The penalty p_λ in (5.8) ranges from quadratic to linear based on the value of λ. Keeping the penalty partly linear ($\lambda > 0$) helps gradient descent drive the overlap

loss to *exactly* zero because a purely quadratic loss would result in a vanishing derivative when overlap constraints approach satisfaction.

Running the Minimization in Practice

When minimizing (5.7) using SGD, we initially place cluster centers randomly within a sphere. The volume V of this sphere influences the initial overlaps between clusters. To select an appropriate value, we fix the ratio ρ of the sum of cluster volumes to V; essentially, ρ is the density of clusters within the sampling volume. Values of ρ around 10% work well in low dimensions. In higher dimensions, however, results from the mathematics of sphere-packing motivate a downward adjustment. A lower bound by Ball, 1992 states that the maximum achievable density when placing non-overlapping spheres inside $\mathbb{R}^p$ is at least $p2^{1-p}$. Thus, in p dimensions we use an adjusted density ρ^{adj} defined by

$$\rho^{\text{adj}}(p) = p2^{1-p}\rho^{2D},$$

where $\rho^{2D} \approx 10\%$ is the equivalent density in 2D.

Following initialization, we optimize the cluster centers $\{\boldsymbol{\mu}_i\}_{i=1}^k$ using stochastic gradient descent. During this process, the principal axes and their lengths are fixed. Each iteration performs the update

$$\left[\boldsymbol{\mu}_1|\boldsymbol{\mu}_2|\ldots|\boldsymbol{\mu}_k\right] \leftarrow \left[\boldsymbol{\mu}_1|\boldsymbol{\mu}_2|\ldots|\boldsymbol{\mu}_k\right] - \eta\left[\nabla_{\boldsymbol{\mu}_1}\ell_i|\nabla_{\boldsymbol{\mu}_2}\ell_i|\ldots|\nabla_{\boldsymbol{\mu}_k}\ell_i\right] \tag{5.9}$$

on the single-cluster loss ℓ_i, where η is the learning rate. For each epoch, we randomly permute the order $i = 1, 2, \ldots, k$ of clusters and apply the updates (5.9) in turn for each cluster.

Experiments suggest that our minimization procedure drives the overlap loss to zero at an exponential rate (linear convergence rate), as expected for gradient descent. The number of epochs required seems largely independent of the number of clusters, though it increases slightly with the number of dimensions.

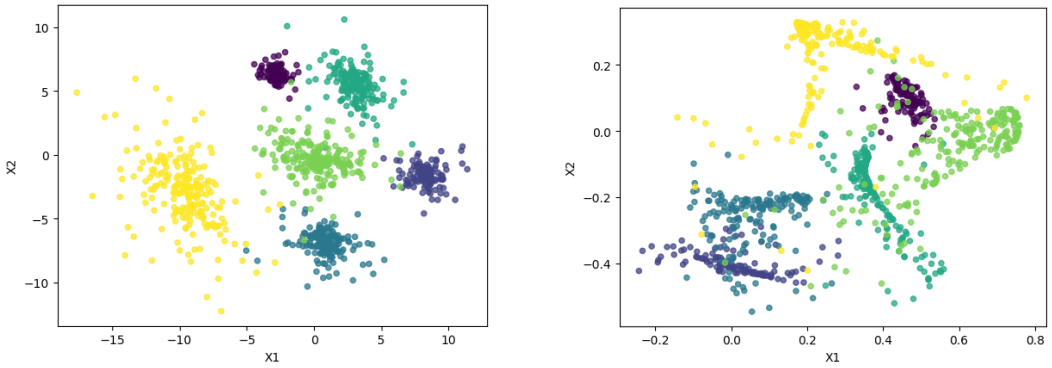
Making Cluster Shapes More Irregular

In many application domains, ellipsoidal center-based clusters are good models for the data. This is often the case when clusters can be characterized by a "typical" element. For example, in single-cell RNA sequencing, Chen et al., 2020 represent the differences in transcriptional activity across cell types using Gaussian mixture models. In this case, the cluster centers correspond to each cell type's typical gene

expression profile. Similarly, in deep learning, a large language model's hidden states often form compact clusters that can effectively classify text (Howard and Ruder, 2018), assess the truthfulness of a response (Azaria and Mitchell, 2023b), and help detect out-of-distribution inputs (Ren et al., 2023b).

However, convex clusters can be inappropriate. For example, Ester et al., 1996 developed the density-based DBSCAN algorithm specifically to handle spatial (GPS) data. Characterized by thin loops and irregular shapes, such data does not fit a center-based clustering paradigm at all. Another example is directional data lying on a sphere (Banerjee et al., 2005; Salah and Nadif, 2019). In this case, the clusters have centers but are not convex.

To accommodate use cases where ellipsoidal clusters are inappropriate, `repliclust` provides two post-processing functions for creating non-convex clusters. The first, `repliclust.distort`, passes a data set through a randomly initialized neural network, as shown in Figure 5.5. The second, `repliclust.wrap_around_sphere`, wraps a p-dimensional data set around the $(p+1)$-dimensional sphere by applying an inverse stereographic projection, as shown in Figure 5.6. Appendix D describes the default architecture of the neural network used in `repliclust.distort`.

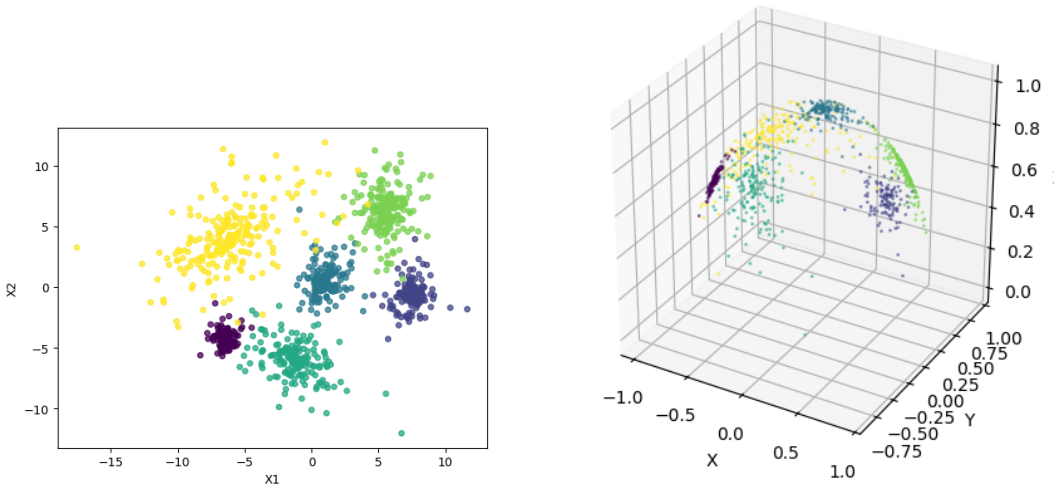


(a) Before `distort` (b) After `distort`

Figure 5.5: You can create non-convex, irregularly shaped clusters by applying the `distort` function, which runs your dataset through a randomly initialized neural network.

Generating Archetypes from Natural Language

To fully realize the potential of a high-level, archetype-based approach to synthetic data generation, our package `repliclust` draws on the OpenAI API (OpenAI, 2023) to allow users to create data set archetypes directly from verbal descriptions in English.



(a) Before `wrap_around_sphere` (b) After `wrap_around_sphere`

Figure 5.6: You can create directional data by wrapping your dataset around the sphere using the `wrap_around_sphere` function. The function works in arbitrary dimensions.

To map a user's natural language input to the high-level geometric parameters of Table 5.1, we use few-shot prompting of OpenAI's GPT-4o large language model model (Brown et al., 2020a; OpenAI, 2024b). We use this same approach to automatically create short identifiers for archetypes, such as

"ten_highly_oblong_very_different_shapes_moderate_overlap."

These identifiers contain no white space and obey the naming rules for Python variables.

Version 1.0.0. of `repliclust` uses 22-shot prompting to map archetype descriptions to parameters, and 11-shot prompting to generate archetype identifiers, which appears to work well. In the rare case that the natural language workflow fails, `repliclust` throws an error. We release all few-shot examples and prompt templates in the `natural_language` module of our code base. They are also reproduced in Appendix F.

5.4 Results

In this section, we first illustrate the convenience of archetype-based data generation by conducting a mock benchmark between several clustering algorithms. Second, we verify that our notion of cluster overlap effectively captures clustering difficulty

(even in higher dimensions), and study the distribution of pairwise overlaps in data sets with multiple clusters.

Mock Benchmark

We demonstrate the convenience and informativeness of our archetype-based data generator by running a mock benchmark comparing several clustering algorithms on data drawn from different archetypes. The benchmark is not meant to comprehensively evaluate the strengths and weaknesses of the clustering algorithms we consider. Instead, our goal is to demonstrate the advantages (in convenience and informativeness) of an archetype-based approach to synthetic data generation.

First, we construct six archetypes by providing `repliclust` with the following verbal descriptions:

1. `'twelve clusters of different distributions'`
2. `'twelve clusters of different distributions and high class imbalance'`
3. `'seven highly separated clusters in 10D with very different shapes'`
4. `'seven clusters in 10D with very different shapes and significant overlap'`
5. `'four clusters in 100D with 100 samples each'`
6. `'four clusters in 100D with 1000 samples each'`

`Repliclust` maps these descriptions to the following data set archetypes:

1. ```
   {'name': 'twelve_clusters_different_distributions',
       'n_clusters': 12, 'dim': 2, 'n_samples': 1200,
       'aspect_ref': 1.5, 'aspect_maxmin': 2, 'radius_maxmin': 3,
       'imbalance_ratio': 2, 'max_overlap': 0.05, 'min_overlap':
       0.001, 'distributions': ['normal', 'exponential', 'gamma',
       'weibull', 'lognormal']}
   ```
   ```

2. {‘name’: ‘twelve_different_distributions_high_class_imbalance’,
   ‘n_clusters’: 12, ‘dim’: 2, ‘n_samples’: 1200,
   ‘aspect_ref’: 1.5, ‘aspect_maxmin’: 2, ‘radius_maxmin’: 3,
   ‘imbalance_ratio’: 5, ‘max_overlap’: 0.05, ‘min_overlap’:
   0.001, ‘distributions’: [‘normal’, ‘exponential’, ‘gamma’,
   ‘weibull’, ‘lognormal’]}

3. {‘name’: ‘seven_highly_separated_10d_very_different_shapes’,
   ‘n_clusters’: 7, ‘dim’: 10, ‘n_samples’: 700, ‘aspect_ref’:
   1.5, ‘aspect_maxmin’: 3.0, ‘radius_maxmin’: 3.0,
   ‘imbalance_ratio’: 2, ‘max_overlap’: 0.0001, ‘min_overlap’:
   1e-05, ‘distributions’: [‘normal’, ‘exponential’]}

4. {‘name’: ‘seven_very_different_shapes_significant_overlap_10d’,
   ‘n_clusters’: 7, ‘dim’: 10, ‘n_samples’: 700, ‘aspect_ref’:
   1.5, ‘aspect_maxmin’: 3.0, ‘radius_maxmin’: 3,
   ‘imbalance_ratio’: 2, ‘max_overlap’: 0.2, ‘min_overlap’:
   0.1, ‘distributions’: [‘normal’, ‘exponential’]}

5. {‘name’: ‘four_clusters_100d_100_samples_each’, ‘n_clusters’:
   4, ‘dim’: 100, ‘n_samples’: 400, ‘aspect_ref’: 1.5,
   ‘aspect_maxmin’: 2, ‘radius_maxmin’: 3, ‘imbalance_ratio’:
   2, ‘max_overlap’: 0.05, ‘min_overlap’: 0.001,
   ‘distributions’: [‘normal’, ‘exponential’]}

6. {‘name’: ‘four_clusters_100d_1000_samples_each’, ‘n_clusters’:
   4, ‘dim’: 100, ‘n_samples’: 4000, ‘aspect_ref’: 1.0,
   ‘aspect_maxmin’: 1.0, ‘radius_maxmin’: 3,
   ‘imbalance_ratio’: 2, ‘max_overlap’: 0.05, ‘min_overlap’:
   0.001, ‘distributions’: [‘normal’, ‘exponential’]}

Note that the automatically generated names for archetype 5. and 6. did not end in the suffix “_each.” We added this suffix for clarity.

Examining the Generated Data Sets

Figure 5.7 shows four representative data sets with convex clusters drawn from each archetype. Figure 5.8 shows non-convex clusters resulting from applying the `distort` function (as described in Section 5.3). For archetypes with dimensionality

greater than two, we use t-SNE to visualize the data sets in 2D (Maaten and Hinton, 2008).

The figures show that `repliclust` effectively generates data sets with similar geometric characteristics. Moreover, the data sets match up well with the user-specified verbal descriptions. Applying `distort` seems to make some clusters very long and thin, but otherwise results in satisfying non-convex clusters. The `distort` function seems to roughly preserve cluster overlaps; we leave a careful study of this to future work. The 10 and 100-dimensional archetypes indicate that our overlap control effectively handles different numbers of dimensions.

Comparing the Clustering Algorithms

We compare the following clustering algorithms: K-Means, hierarchical (with Ward linkage), spectral (with radial-basis function affinity), HDBSCAN, and expectation maximization for a Gaussian mixture model (EM-GMM). We originally intended to include DBSCAN as well. However, the heuristics we tried for choosing the neighborhood radius (including what is suggested in Sander et al., 1998) did not work well across the range of dimensionalities we consider, resulting in too many noise points (see also the discussion by Schubert et al., 2017).

We measure performance in terms of adjusted mutual information (AMI) and adjusted Rand index (ARI), based on the ground truth cluster labels (Vinh, Epps, and Bailey, 2010; Hubert and Arabie, 1985). To carry out the benchmark, we sample 10 times from each archetype, resulting in 60 distinct data sets. We repeat this process twice to evaluate performance on convex and non-convex clusters (the latter resulting from applying the `distort` function on new data sets). Thus, the full benchmark is based on 120 distinct data sets. Table 5.2 shows the results for K-Means, hierarchical, spectral, and EM-GMM. All these algorithms receive the true number of clusters as an input.

Table 5.3 separately lists the performance for HDBSCAN, which we ran using the scikit-learn implementation (Pedregosa et al., 2011) with `min_samples=5`. We present the HDBSCAN results separately for two reasons. First, as a density-based algorithm, HDBSCAN cannot make use of the true number of clusters, putting it at a disadvantage. Second, HDBSCAN reports a large number of noise points ($\approx$60% on average), so that its results are not directly comparable to that of the other algorithms. We report the performance of HDBSCAN based only on the non-noise

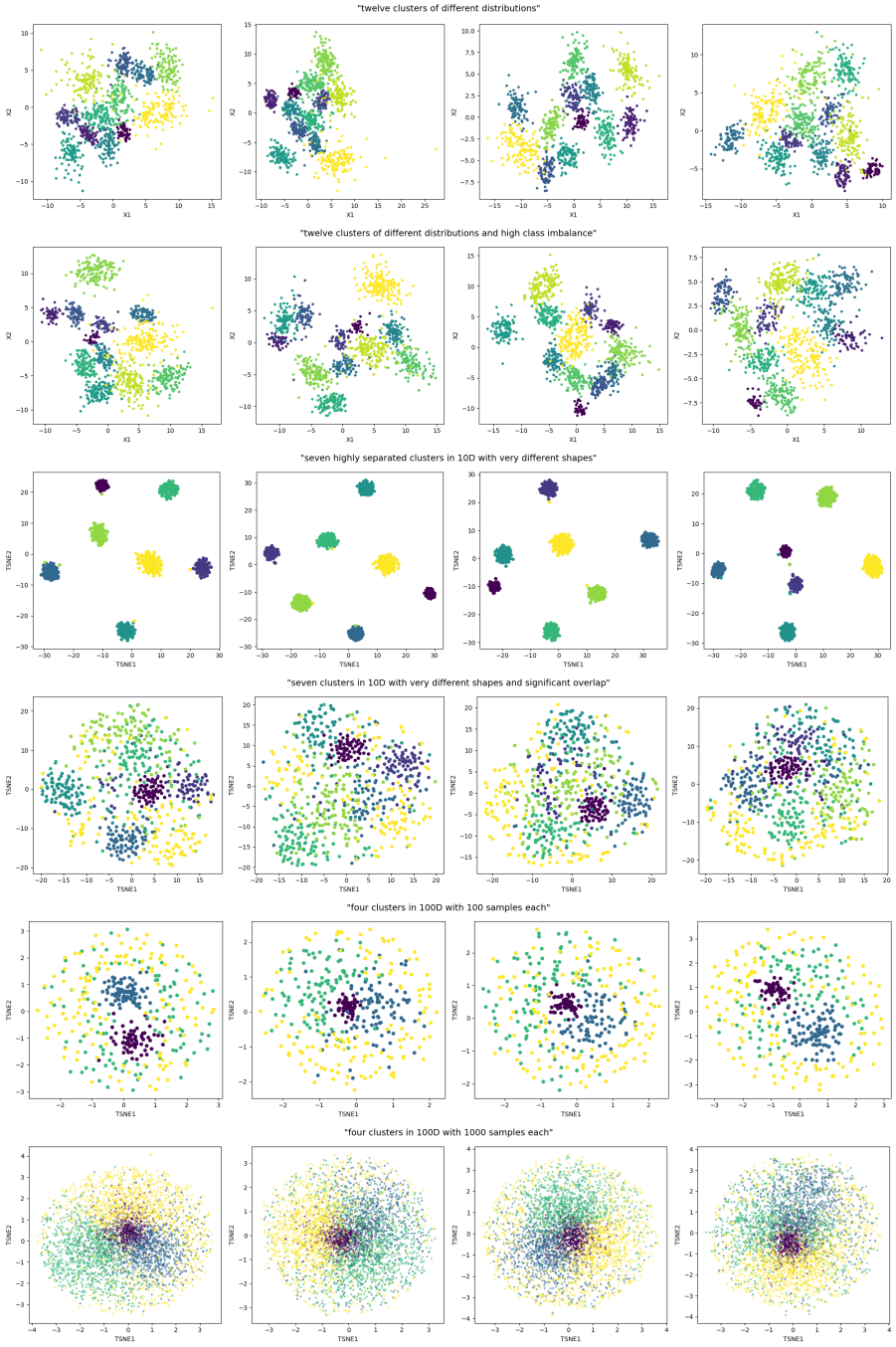


Figure 5.7: Representative convex clusters drawn from the archetypes in our benchmark (not cherry-picked).

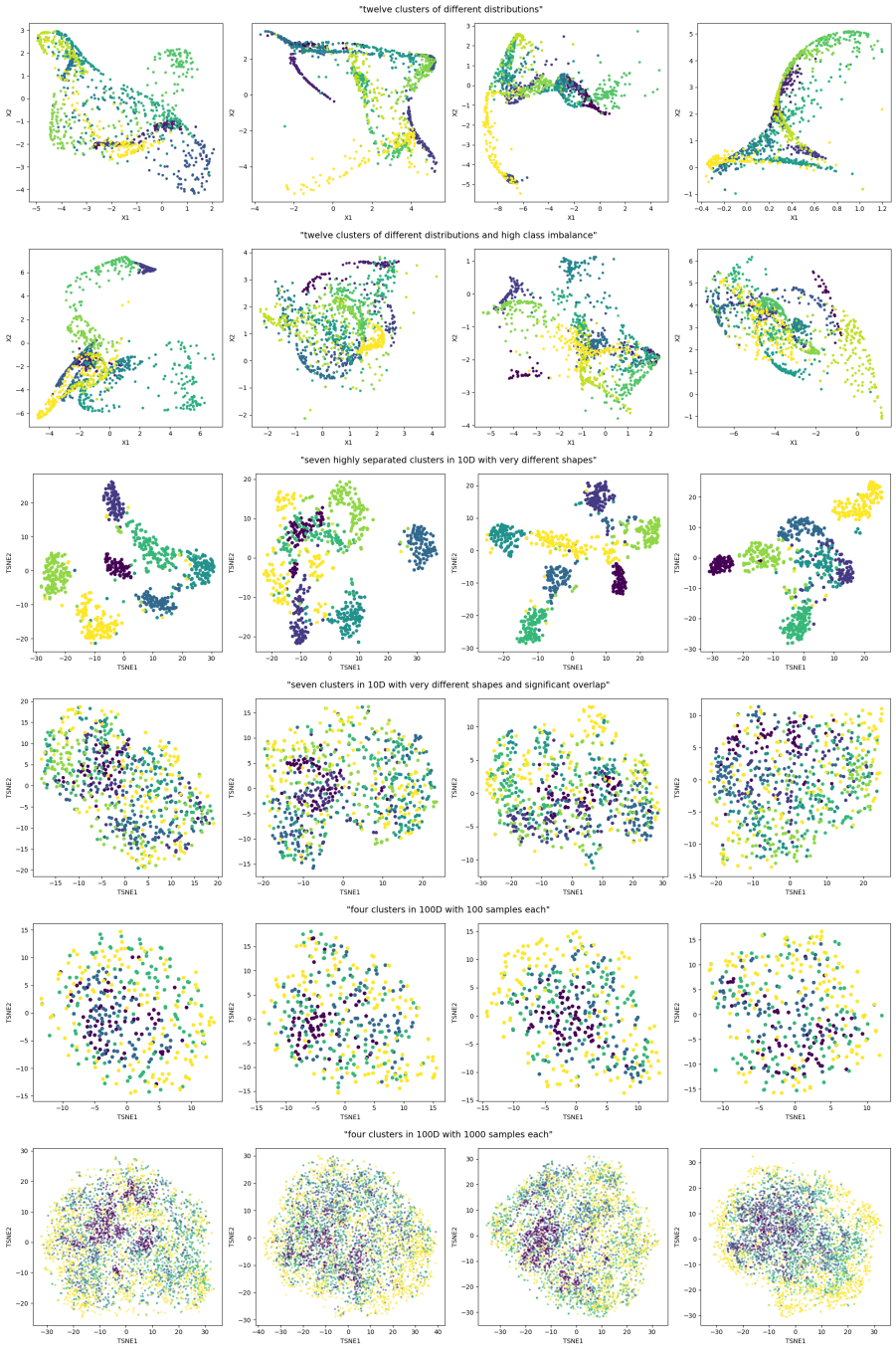


Figure 5.8: Representative non-convex clusters drawn from the archetypes in our benchmark (not cherry-picked).

points. This leads to higher performance numbers, which may compensate for the disadvantage HDBSCAN faces in not knowing the true number of clusters.

Table 5.2: Benchmark results on convex and non-convex data. The best performance for each archetype is printed in **bold**.

Archetype	**Convex**				**Non-Convex**			
	AMI				**AMI**			
	EM-GMM	K-Means	Spectral	Hierarchical	EM-GMM	K-Means	Spectral	Hierarchical
twelve_clusters_different_distributions	**0.884 (.007)**	0.848 (.008)	0.859 (.010)	0.853 (.007)	**0.576 (.017)**	0.563 (.012)	0.565 (.015)	0.565 (.011)
twelve_different_distributions_high_class_imbalance	**0.854 (.008)**	0.836 (.007)	0.852 (.010)	0.834 (.009)	0.544 (.021)	0.542 (.023)	0.547 (.025)	**0.553 (.024)**
seven_highly_separated_10d_very_different_shapes	0.983 (.009)	0.976 (.005)	0.986 (.003)	**0.990 (.002)**	**0.750 (.035)**	0.606 (.027)	0.709 (.027)	0.637 (.029)
seven_very_different_shapes_significant_overlap	0.349 (.019)	**0.452 (.009)**	0.443 (.014)	0.380 (.010)	**0.173 (.014)**	0.149 (.008)	0.165 (.010)	0.145 (.010)
four_clusters_100d_100_samples_each	0.063 (.008)	0.512 (.028)	0.074 (.023)	**0.654 (.022)**	0.077 (.011)	**0.100 (.012)**	0.079 (.008)	0.092 (.008)
four_clusters_100d_1000_samples_each	0.548 (.061)	0.664 (.023)	0.205 (.014)	**0.868 (.025)**	**0.307 (.030)**	0.145 (.014)	0.109 (.012)	0.132 (.016)
Average	0.614 (.044)	0.715 (.025)	0.570 (.046)	**0.763 (.026)**	**0.404 (.032)**	0.351 (.030)	0.362 (.033)	0.354 (.031)
	ARI				**ARI**			
	EM-GMM	K-Means	Spectral	Hierarchical	EM-GMM	K-Means	Spectral	Hierarchical
twelve_clusters_different_distributions	**0.855 (.011)**	0.795 (.012)	0.807 (.018)	0.798 (.014)	0.368 (.021)	**0.365 (.015)**	0.364 (.015)	0.362 (.012)
twelve_different_distributions_high_class_imbalance	**0.800 (.019)**	0.760 (.012)	0.787 (.020)	0.756 (.017)	0.341 (.022)	0.347 (.021)	0.345 (.025)	**0.360 (.024)**
seven_highly_separated_10d_very_different_shapes	0.967 (.020)	0.977 (.005)	0.988 (.003)	**0.992 (.002)**	**0.684 (.045)**	0.505 (.035)	0.628 (.036)	0.523 (.039)
seven_very_different_shapes_significant_overlap	0.238 (.023)	0.375 (.012)	**0.385 (.016)**	0.291 (.015)	**0.109 (.011)**	0.090 (.006)	0.089 (.007)	0.081 (.007)
four_clusters_100d_100_samples_each	0.004 (.007)	0.404 (.036)	0.053 (.014)	**0.554 (.028)**	0.018 (.006)	**0.062 (.011)**	0.053 (.008)	0.051 (.005)
four_clusters_100d_1000_samples_each	0.408 (.048)	0.601 (.040)	0.185 (.011)	**0.874 (.032)**	**0.252 (.029)**	0.088 (.014)	0.065 (.012)	0.072 (.016)
Average	0.545 (.047)	0.652 (.030)	0.534 (.044)	**0.711 (.031)**	**0.295 (.029)**	0.243 (.023)	0.264 (.025)	0.241 (.025)

Table 5.3: Benchmark results for HDBSCAN on convex and non-convex data. Numbers that outperform the algorithms in Table 5.2, with less than 40% noise points, are printed in **bold**.

Archetype	**Convex**			**Non-Convex**		
	AMI	**ARI**	$\mathbf{p_{noise}}$	**AMI**	**ARI**	$\mathbf{p_{noise}}$
twelve_clusters_different_distributions	0.796 (.045)	0.705 (.066)	0.269 (.033)	**0.605 (.016)**	**0.400 (.026)**	0.353 (.007)
twelve_different_distributions_high_class_imbalance	0.733 (.054)	0.572 (.083)	0.238 (.042)	**0.560 (.024)**	**0.356 (.036)**	0.346 (.016)
seven_highly_separated_10d_very_different_shapes	**0.992 (.006)**	0.984 (.014)	0.175 (.014)	0.747 (.106)	0.700 (.117)	0.417 (.043)
seven_very_different_shapes_significant_overlap	0.595 (.110)	0.617 (.124)	0.909 (.020)	0.163 (.050)	0.149 (.066)	0.744 (.072)
four_clusters_100d_100_samples_each	1.000 (.000)	1.000 (.000)	1.000 (.000)	0.794 (.137)	0.804 (.131)	0.990 (.007)
four_clusters_100d_1000_samples_each	0.800 (.133)	0.800 (.133)	0.999 (.000)	0.031 (.018)	0.023 (.016)	0.687 (.070)
Average	0.819 (.035)	0.780 (.040)	0.599 (.049)	0.483 (.047)	0.405 (.047)	0.590 (.036)

The results show that hierarchical clustering performs best on the convex cluster shapes, as long as there is sufficient separation between clusters. On the non-convex clusters, EM-GMM exhibits the strongest performance even though the clusters are no longer multivariate normal after applying `distort` (see Figure 5.8).

K-Means and hierarchical clustering both hold up well on the high-dimensional data, including in the low-sample regime of 100 samples per cluster in 100D. By contrast, EM-GMM dramatically benefits from more samples on the high-dimensional data.

Spectral clustering does not display competitive performance in this benchmark. While it improves over K-Means in some scenarios, it delivers weaker performance in high dimensions and never performs best across all algorithms.

HDBSCAN shows great difficulty in handling high dimensionality or significant overlap between clusters. However, the algorithm shows strong performance on the 12 non-convex clusters drawn from diverse distributions. We note again that HDBSCAN does not have access to the true number of clusters, in contrast with the other algorithms.

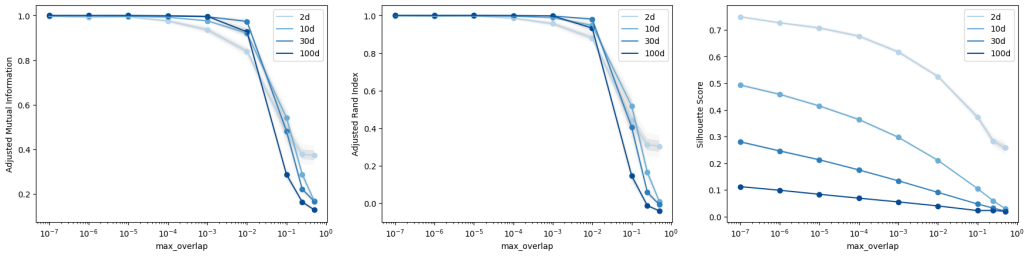
Minimax Classification Error Captures Clustering Difficulty

In Section 5.3, we defined the overlap between two clusters in terms of the error rate of the best minimax linear classifier. We verify that this notion of overlap conveys clustering difficulty by measuring clustering performance on data sets with different degrees of overlap. For this simulation, we consider data sets with two clusters drawn from an archetype we described as "two clusters with very different shapes in pD". This verbal description yields an archetype with parameters

```
{ 'name': 'two_very_different_shapes_|$p$|d', 'n_clusters': 2,
    'dim': |$p$|, 'n_samples': 200, 'aspect_ref': 1.5,
    'aspect_maxmin': 3.0, 'radius_maxmin': 3, 'imbalance_ratio': 2,
    'max_overlap': 0.05, 'min_overlap': 0.001, 'distributions':
    ['normal', 'exponential'] },
```

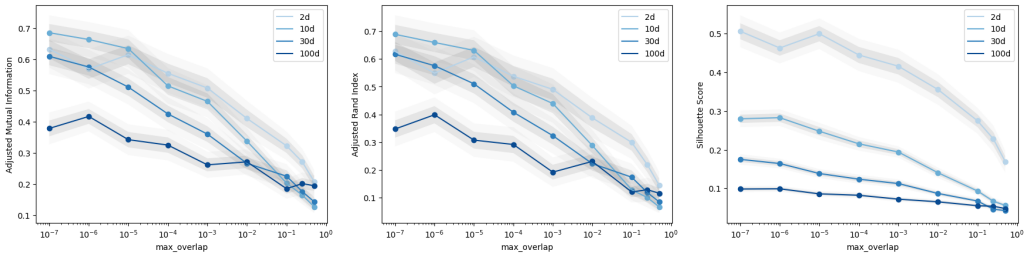
where the dimensionality p ranges across $[2, 10, 30, 100]$. We vary `max_overlap` from 10^{-7} to 0.5, while setting `min_overlap` = `max_overlap`/10. For each overlap setting, we generate 100 distinct data sets and evaluate the average clustering performance of hierarchical clustering, quantified in terms of adjusted mutual information (AMI) and adjusted Rand index (ARI), as in the benchmark of Section 5.4. We choose hierarchical clustering because it is computationally efficient and performed well in the benchmark. We repeat this process twice, where in the second run we make clusters non-convex by applying the `distort` function.

Figure 5.9 confirms that clustering difficulty rises with increasing overlap. Figure 5.10 shows the same in the case of non-convex clusters, suggesting that applying `distort` maintains the desired relationship between overlap and clustering difficulty. Additionally, both figures show how our cluster overlap relates to the silhouette score, a popular metric for quantifying clustering difficulty (Rousseeuw, 1987; Shand et al., 2019). At a fixed value of `max_overlap`, the silhouette score decreases markedly with a rise in dimensionality. This is not an artifact of our overlap measure, since plotting clustering performance vs silhouette score shows a similar dependence on dimensionality (not shown). This makes sense since the silhouette score is based on the difference of Euclidean distances, and distances between points tend to become more similar in high dimensions (Aggarwal, Hinneburg, and Keim, 2001).



(a) AMI vs Overlap (b) ARI vs Overlap (c) Silhouette vs Overlap

Figure 5.9: Cluster overlap predicts clustering difficulty for convex clusters. Clustering performance is measured in terms of adjusted mutual information (AMI, left) and adjusted Rand index (ARI, middle). Right: the silhouette score is more sensitive to dimensionality but otherwise aligns well with our cluster overlap.



(a) AMI vs Overlap (b) ARI vs Overlap (c) Silhouette vs Overlap

Figure 5.10: Cluster overlap predicts clustering difficulty for non-convex clusters. Clustering performance is measured in terms of adjusted mutual information (AMI, left) and adjusted Rand index (ARI, middle). Right: the silhouette score is more sensitive to dimensionality but otherwise aligns well with our cluster overlap.

Examining the Distribution of Pairwise Overlaps

Since `repliclust` controls cluster overlap on the level of entire data sets by setting two global parameters (`max_overlap` and `min_overlap`), it is worthwhile to investigate the distributions of pairwise overlaps on datasets with multiple clusters. Figure 5.11 shows the distribution of pairwise overlaps for six data set archetypes, confirming that setting `max_overlap` and `min_overlap` effectively controls the pairwise overlaps between clusters.

5.5 Related Work

Simulations on synthetic data play an important role in cluster analysis. Accordingly, many synthetic data generators for cluster analysis have been proposed. However, the key idea in this paper is to specify the overall geometric characteristics of synthetic data in a high-level manner via data set archetypes. By contrast, previous data generators have put the onus on the user to *design* the overall geometric structure by

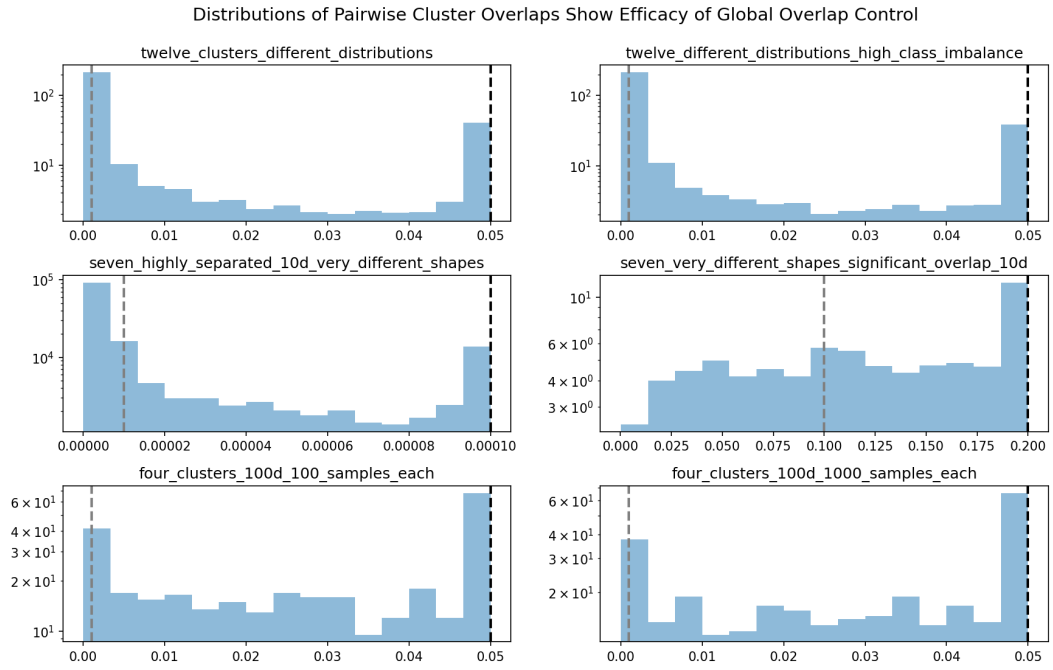


Figure 5.11: Distributions of pairwise overlaps between clusters reveal that global overlap control works well. The black dashed line indicates the `max_overlap` setting, and the gray dashed line indicates `min_overlap`. Note that `min_overlap` is not a lower bound on pairwise overlap because it only requires that each cluster share the minimum degree of overlap with *at least one* other cluster; overlaps with other clusters may be lower.

carefully tuning lower-level properties of individual clusters.

In the following overview, we mainly focus on general-purpose generators. However, the literature has also proposed more specialized solutions to fill specific needs in the community. For example, Beer, Schüler, and Seidl, 2019 present a data generator for subspace clustering, Gan and Tao, 2015 evaluates density-based clustering using a data generation process based on random walks, and Handl and Knowles, 2005 focus on creating long and thin ellipsoidal clusters in higher dimensions. None of these contributions focus on giving high-level control over the overall geometry of the data sets.

Milligan and Cooper, 1985 implements a generator for generating several clusters in up to 8 dimensions. The method enforces an upper bound on cluster overlap by limiting overlap in the first dimension, but does not otherwise provide control over high-level geometric structure.

Pei and Zaïane, 2006 present a compelling software system for generating two-dimensional clusters, which creates data sets with specified clustering difficulty

"easy," "medium," or "hard"; "easy" data consists only of spherical/convex clusters, whereas "medium" and "hard" data include curve segments and special shapes like letters of the alphabet. The generator does not offer high-level control over data set geometry except for the difficulty scale and the density of noise points to add to the data. The software comes with an appealing graphical user interface.

The popular scikit-learn library for machine learning (Pedregosa et al., 2011) offers several functions for creating synthetic clusters. Among these, some are aimed at reproducing canonical 2D toy data sets like concentric circles and nested moons (`make_moons`, `make_circles`), while others focus on sampling multivariate normal clusters. These functions offer some valuable features, such as the ability to create datasets with informative and redundant features, as in the `make_classification function`. However, they do not control overall geometric characteristics of the data.

The data mining framework ELKI (Schubert and Zimek, 2019) provides a synthetic data generator based on specifying probabilistic models in XML files. This XML-based approach makes it easy to reproduce benchmarks by sharing the underlying XML files. Drawing inspiration from this work, we have implemented an `Archetype.describe` function allowing users to easily share collections of data set archetypes as JSONL files.

The generators OCLUS (Steinley and Henson, 2005) and GenRandomClust (Qiu and Joe, 2006) focus on providing more sophisticated overlap control compared to previous generators. GenRandomClust extends the generator of Milligan and Cooper, 1985 by managing overlaps between clusters with different ellipsoidal shapes and arbitrary spatial orientations. Similar to our classification error-based notion of cluster overlap, their method finds an optimal separation direction between two clusters. To enforce the desired degree of overlap, the algorithm initially places cluster centers on a scaffold, then scales individual edges of the scaffold up or down to meet the overlap constraint. The method supports making *all* cluster shapes more or less elongated, but does not otherwise provide high-level control over data set geometry. Moreover, the scaling operations undertaken to manage cluster overlaps implicitly sacrifice control over cluster volumes (which, in `repliclust`, can be managed independently).

OCLUS (Steinley and Henson, 2005) quantifies cluster overlap in terms of the shared density between two clusters. The generator uses analytical formulas for integrals of several interesting probability distributions (including exponential, gamma, and

chi-square), thereby effectively managing overlaps between non-normal clusters. As a result, the method is limited to treating all dimensions independently, so that cluster distributions simplify as the products of marginals. The paper contains a valuable discussion of the distinction between marginal and joint cluster overlaps (of which their software supports both).

Like other existing generators, OCLUS does not aspire to helping the user establish the overall geometric characteristics of synthetic data sets. To generate a data set, the user must provide a covariance matrix for each cluster, the desired overlaps between all pairs of clusters, and a design matrix specifying which clusters overlap at all. In sum, generating 10 clusters in 10D requires supplying over 500 numbers, compared to only a handful in `repliclust` (or none if the user chooses to describe the archetype in English).

MDCGen (Iglesias et al., 2019 is a feature-rich generator that supports many desiderata in cluster analysis, such as overlap control, different probability distributions, subspace clusters, and the ability to add noise points. In particular, it is nice to be able to place noise points away from the clusters, which is made possible by the grid-based strategy for placing cluster centers. MDCGen does not target the overall geometric characteristics of synthetic data sets, instead giving users low-level control enabling extensive configurability. For example, managing the overlap between clusters involves setting compactness coefficients, grid granularity, and overall scale, compared to only tweaking `max_overlap` in `repliclust` (`min_overlap` may also have to be tweaked but it can usually stay at `max_overlap`/10 or a similar value). In the words of the authors, "to enable covering a broad range of dataset possibilities, the parameters are multiple and some training for tuning the tool is required."

Finally, the HAWKS generator (Shand et al., 2019) controls cluster overlaps using an evolutionary algorithm that evolves the means and covariance matrices of multivariate normal distributions. The paper applies this framework to create data sets with a user-specified silhouette score representing clustering difficulty (Rousseeuw, 1987). In principle, the evolutionary framework can be extended to attain desired high-level geometric characteristics. Of these, the authors consider two examples, cluster overlaps and elongations (the latter relating to our notion of cluster aspect ratio, as listed in Table 5.1). An interesting aspect of HAWKS is the ability to generate data sets that maximize the performance difference between two clustering algorithms. This feature is especially useful in two dimensions, since we can then visually examine the data sets to better understand when each algorithm succeeds

or fails.

5.6 Conclusion

In this paper, we have presented an archetype-based approach to synthetic data generation for cluster analysis. Our method works by summarizing the overall geometric characteristics of a probabilistic mixture model with a few high-level parameters, and sampling mixture models subject to these constraints. To convey the convenience and informativeness of such an archetype-based approach, we have implemented a natural language interface allowing users to create archetypes purely from verbal descriptions in English. Thus, our software `repliclust` makes it possible to run an entire benchmark by describing the desired evaluation scenarios in English.

Although our data generator relies on creating a skeleton of convex, ellipsoidal clusters, we have implemented ways to make the cluster shapes more irregular and complex. The first method passes convex clusters through a randomly initialized neural network, making their shapes non-convex and irregular. The second method creates directional datasets by wrapping p-dimensional convex clusters around the $(p + 1)$-dimensional sphere.

In future work, we are most interested in learning data set archetypes that mimic the geometric characteristics of real data. In application domains where multivariate Gaussians provide a good model for the data, it would suffice to fit a Gaussian mixture model, empirically measure its high-level geometric parameters (as listed in Table 5.1), then create a synthetic data set archetype with these parameters. For application domains with non-convex clusters, it would be interesting to implement a generalized version of this approach. We can imagine training an auto-encoder type neural network that maps non-convex clusters into a hidden space where they become multivariate Gaussian, then undoes the transformation. Defining an archetype based on the high-level geometric characteristics in the hidden space would presumably allow us to sample irregularly shaped clusters that look similar to those found in real data.

Chapter 6

CONCLUSION

6.1 Summary

This thesis has covered substantial ground. Starting from the recognition that deploying AI in practice still poses many challenges, we focused on three particular pain points: 1) selecting the most suitable AI models, 2) balancing the economics of their usage, and 3) further reducing AI models' error rates by flagging difficult queries for review by human experts.

We noted the need for a unified framework addressing these challenges, and presented several components of such a framework:

Uncertainty-Aware System Optimization. Taking the example of LLM cascades, we presented a Markov-copula probabilistic model for the joint distribution of the uncertainties of several LLMs, providing a foundation for continuously optimizing systems of AI models while explicitly modeling uncertainty. Compared to Bayesian optimization, our methodology yields improved error-cost curves for cascades with more than two LLMs, as measured by an up to 7.2% percentage decrease in the area under the curve. Notably, our method is more data-efficient, with the performance gap rising up to 16.5% for $n \leq 30$ labeled calibration samples.

Economic Evaluation. Framing LLM systems as reward-seeking agents, we presented an economic formalism for selecting *the* best AI model when optimizing multiple conflicting performance objectives. We applied this methodology to evaluate the utility of less powerful AI models for automating meaningful human work. Surprisingly, we found that practitioners should deploy only the most powerful AI models as soon as the cost of an AI error exceeds about $1. In addition, single models outperform cascades on risk-sensitive domains, where each mistake costs at least $100. These results provide informative guidance for adopting AI with commercial considerations in mind.

Error Reduction with Human-in-the-Loop. By giving large reasoning models the ability to abstain from difficult queries (and send them to a human expert), we unlocked additional accuracy gains for large reasoning LLMs, for example, lowering the error rate of Qwen3 235B-A22B from 3% to below 1% on competition-level

mathematics problems when rejecting around 10% of queries. In addition, to speed up reasoning models' response times, we explored fronting them with a faster non-reasoning model. Implementing this strategy, we attained latency reductions of around 30% at the cost of a 3% impairment in accuracy. Unfortunately, we identified *latency drag* as a headwind preventing larger gains.

Proof-of-Concept System for Synthetic Data Generation. Finally, we illustrated the practical potential of LLM systems by implementing a natural language-based synthetic data generator for cluster analysis. This approach enables statistical researchers to create benchmarking scenarios purely by describing them in English, making synthetic data benchmarks significantly more interpretable and generalizable. Leveraging our software to compare several popular clustering algorithms, we identified distinct strengths and weaknesses based on the overall geometry of the synthetic data sets.

6.2 Remaining Challenges

Many challenges remain in adopting AI models for practical use cases. We consider two problems of special interest: 1) making LLM systems robust to distribution shifts, and 2) carefully profiling human abilities.

Robustness to Distribution Shifts. One weakness of most LLM systems consists of the necessity to re-tune the system for different use cases. For example, for a cascade, the same deferral thresholds ensuring a low error rate on domain A might be entirely insufficient to yield good performance on a different domain B. This problem is highly relevant for customer-facing LLM deployments, since the LLM's query distribution is entirely up to the customer. Customers may abruptly send different queries, leading to a sharp distribution shift.

This problem may be addressed in at least two different ways: first, domain-agnostic "universal" calibration of LLM confidence may obviate the need for re-tuning confidence thresholds after distribution shifts. The work by Shen et al. (2024) is an effort in this direction. However, achieving universal calibration across domains remains a challenging problem.

Second, an online approach may be taken to interactively learn an AI model's confidence. By gradually adapting confidence calculations based on incoming data, this strategy manages query distribution shifts by *leaning into them*. As an additional benefit, this approach reduces the need for labeled calibration data when initially deploying the system. Zhang et al. (2024b) explore this direction by framing cascade

tuning as a reinforcement learning problem. From a practical point-of-view, the key problem consists of the rapidity with which the system can adjust to shifts in the incoming data stream. Hence, strategies to shorten the system's response time, as in engineering control theory, would be of significant practical interest.

Profiling of Human Abilities. As artificial intelligence becomes more and more capable, carefully profiling the abilities of humans is increasingly important. For example, a human-in-the-loop system in which human experts answer the most difficult queries can only work if we trust human judgment in these cases. As the supremacy of human over artificial intelligence crumbles, a complementary human-AI approach may ultimately prove more fruitful: rather than treating human experts as a fallback for imperfect AI, embedding humans on an equal footing with AI models leads to fully hybrid human-AI systems. Such constellations may leverage the strengths and weaknesses of each participant—whether human or machine. However, seamlessly operating such systems requires adequate knowledge about the boundaries of human expertise.

BIBLIOGRAPHY

Aggarwal, Charu C., Alexander Hinneburg, and Daniel A. Keim (2001). "On the Surprising Behavior of Distance Metrics in High Dimensional Space." In: *Database Theory — ICDT 2001*. Ed. by Jan Van den Bussche and Victor Vianu. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 420–434. ISBN: 978-3-540-44503-6.

Aggarwal, Pranjal et al. (2024). *AutoMix: Automatically Mixing Language Models*. arXiv: 2310.12963 [cs.CL]. URL: https://arxiv.org/abs/2310.12963.

Alibaba AI (2025). *Qwen3 Technical Report*. arXiv: 2505.09388 [cs.CL]. URL: https://arxiv.org/abs/2505.09388.

Amershi, Saleema et al. (Dec. 2014). "Power to the People: The Role of Humans in Interactive Machine Learning." In: *AI Magazine* 35.4, pp. 105–120. DOI: 10.1609/aimag.v35i4.2513. URL: https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/2513.

Anderson, Theodore Wilbur and Raghu Raj Bahadur (1962). "Classification into two Multivariate Normal Distributions with Different Covariance Matrices." In: *The Annals of Mathematical Statistics* 33.2, pp. 420–431.

Azaria, Amos and Tom Mitchell (2023a). *The Internal State of an LLM Knows When It's Lying*. arXiv: 2304.13734 [cs.CL]. URL: https://arxiv.org/abs/2304.13734.

– (2023b). "The Internal State of an LLM Knows When It's Lying." In: *Findings of the Association for Computational Linguistics: EMNLP 2023*. Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. Singapore: Association for Computational Linguistics, pp. 967–976. DOI: 10.18653/v1/2023.findings-emnlp.68. URL: https://aclanthology.org/2023.findings-emnlp.68.

Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E Hinton (2016). "Layer Normalization." In: *arXiv preprint arXiv:1607.06450*.

Ball, Keith (1992). "A lower bound for the optimal density of lattice packings." In: *International Mathematics Research Notices* 1992.10, pp. 217–221. eprint: https://academic.oup.com/imrn/article-pdf/1992/10/217/6767935/1992-10-217.pdf.

Ballon, Marthe, Andres Algaba, and Vincent Ginis (2025). *The Relationship Between Reasoning and Performance in Large Language Models – o3 (mini) Thinks Harder, Not Longer*. arXiv: 2502.15631 [cs.LG]. URL: https://arxiv.org/abs/2502.15631.

Banerjee, Arindam et al. (2005). "Clustering on the Unit Hypersphere using von Mises-Fisher Distributions." In: *Journal of Machine Learning Research* 6.46, pp. 1345–1382. URL: http://jmlr.org/papers/v6/banerjee05a.html.

Banholzer, Stefan and Stefan Volkwein (2019). "Hierarchical Convex Multiobjective Optimization by the Euclidean Reference Point Method." Preprint SPP1962-117.

Beer, Anna, Nadine-Sarah Schüler, and Thomas Seidl (Jan. 2019). "A Generator for Subspace Clusters." In.

Bertsekas, Dimitri P. (1999). *Nonlinear Programming*. 2nd ed. Belmont, MA: Athena Scientific. ISBN: 1886529000.

Blum, A., J. Hopcroft, and R. Kannan (2020). *Foundations of Data Science*. Cambridge University Press. ISBN: 9781108485067. URL: https://books.google.com/books?id=koHCDwAAQBAJ.

Börzsönyi, Stephan, Donald Kossmann, and Konrad Stocker (2001). "The Skyline operator." In: *Proceedings 17th International Conference on Data Engineering*, pp. 421–430. URL: https://api.semanticscholar.org/CorpusID:5812098.

Branke, Jürgen et al., eds. (Oct. 2008). *Multiobjective Optimization. Interactive and Evolutionary Approaches*. Vol. 5252. Lecture Notes in Computer Science. XX + 470 pages. Berlin; Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-540-88907-6. DOI: 10.1007/978-3-540-88908-3. URL: https://doi.org/10.1007/978-3-540-88908-3.

Brown, Tom et al. (2020a). "Language Models are Few-Shot Learners." In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., pp. 1877–1901. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Brown, Tom B. et al. (2020b). "Language Models are Few-Shot Learners." In: arXiv: 2005.14165 [cs.CL]. URL: https://arxiv.org/abs/2005.14165.

Brynjolfsson, Erik, Danielle Li, and Lindsey R Raymond (Apr. 2023). *Generative AI at Work*. Working Paper 31161. National Bureau of Economic Research. DOI: 10.3386/w31161. URL: http://www.nber.org/papers/w31161.

Burns, Collin et al. (2024). *Discovering Latent Knowledge in Language Models Without Supervision*. arXiv: 2212.03827 [cs.CL]. URL: https://arxiv.org/abs/2212.03827.

Campello, Ricardo J. G. B., Davoud Moulavi, and Joerg Sander (2013). "Density-Based Clustering Based on Hierarchical Density Estimates." In: *Advances in Knowledge Discovery and Data Mining*. PAKDD'13. Gold Coast, Australia: Springer-Verlag, pp. 160–172. ISBN: 9783642374555. DOI: 10.1007/978-3-642-37456-2_14. URL: https://doi.org/10.1007/978-3-642-37456-2_14.

Casella, G. and R.L. Berger (2002). *Statistical Inference*. 2nd ed. Duxbury Press, Pacific Grove.

cash cow (2025). *Merriam-Webster.com*. Accessed: 24 July 2025. URL: https://www.merriam-webster.com.

Chen, Chao et al. (2024a). *INSIDE: LLMs' Internal States Retain the Power of Hallucination Detection*. arXiv: 2402.03744 [cs.CL]. URL: https://arxiv.org/abs/2402.03744.

Chen, Lingjiao, Matei Zaharia, and James Zou (2023). *FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance*. arXiv: 2305.05176 [cs.LG]. URL: https://arxiv.org/abs/2305.05176.

Chen, Lingjiao et al. (2024b). "Are More LLM Calls All You Need? Towards the Scaling Properties of Compound AI Systems." In: *Advances in Neural Information Processing Systems*. Ed. by A. Globerson et al. Vol. 37. Curran Associates, Inc., pp. 45767–45790. URL: https://proceedings.neurips.cc/paper_files/paper/2024/file/51173cf34c5faac9796a47dc2fdd3a71-Paper-Conference.pdf.

Chen, Sisi et al. (2020). "Dissecting heterogeneous cell populations across drug and disease conditions with PopAlign." In: *Proceedings of the National Academy of Sciences* 117.46, pp. 28784–28794. DOI: 10.1073/pnas.2005990117. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.2005990117. URL: https://www.pnas.org/doi/abs/10.1073/pnas.2005990117.

Chiang, Wei-Lin et al. (2024). "Chatbot Arena: An Open Platform for Evaluating LLMs by Human Preference." In: *Forty-first International Conference on Machine Learning*. URL: https://openreview.net/forum?id=3MW8GKNyzI.

Chowdhery, Aakanksha et al. (2022). *PaLM: Scaling Language Modeling with Pathways*. arXiv: 2204.02311 [cs.CL]. URL: https://arxiv.org/abs/2204.02311.

Chung, Hyung Won et al. (2022). *Scaling Instruction-Finetuned Language Models*. arXiv: 2210.11416 [cs.LG]. URL: https://arxiv.org/abs/2210.11416.

Cobbe, Karl et al. (2021). "Training Verifiers to Solve Math Word Problems." In: *arXiv preprint arXiv:2110.14168*.

Coello, Carlos A. Coello, Arturo Hern'andez Aguirre, and Eckart Zitzler, eds. (Mar. 2005). *Evolutionary Multi-Criterion Optimization Third International Conference, EMO 2005, Guanajuato, Mexico, March 9–11, 2005, Proceedings*. Vol. 3410. Lecture Notes in Computer Science. eBook ISBN 978-3-540-31880-4. Berlin/Heidelberg: Springer, pp. XVI+912. ISBN: 978-3-540-24983-2. DOI: 10.1007/b106458.

Cormack, R. M. (1971). "A Review of Classification." In: *Journal of the Royal Statistical Society. Series A (General)* 134.3, pp. 321–367. ISSN: 00359238. URL: http://www.jstor.org/stable/2344237 (visited on 09/13/2024).

Cowen-Rivers, Alexander et al. (July 2022). "HEBO: Pushing The Limits of Sample-Efficient Hyperparameter Optimisation." In: *Journal of Artificial Intelligence Research* 74.

Das, Indraneel and J. E. Dennis (1997). "A Closer Look at Drawbacks of Minimizing Weighted Sums of Objectives for Pareto Set Generation in Multicriteria Optimization Problems." In: *Structural Optimization* 14.1, pp. 63–69. DOI: 10.1007/BF01197559.

DeepSeek AI (2025). *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*. arXiv: 2501.12948 [cs.CL]. URL: https://arxiv.org/abs/2501.12948.

Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). "Maximum Likelihood from Incomplete Data Via the EM Algorithm." In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1, pp. 1–22. ISSN: 0035-9246. DOI: 10.1111/j.2517-6161.1977.tb01600.x. eprint: https://academic.oup.com/jrsssb/article-pdf/39/1/1/49117094/jrsssb_39_1_1.pdf. URL: https://doi.org/10.1111/j.2517-6161.1977.tb01600.x.

Dettmers, Tim et al. (2024). "LLM.int8(): 8-bit matrix multiplication for transformers at scale." In: *Proceedings of the 36th International Conference on Neural Information Processing Systems*. NIPS '22. New Orleans, LA, USA: Curran Associates Inc. ISBN: 9781713871088.

Ding, Dujian et al. (2024). "Hybrid LLM: Cost-Efficient and Quality-Aware Query Routing." In: *The Twelfth International Conference on Learning Representations*. URL: https://openreview.net/forum?id=02f3mUtqnM.

Dvijotham, Krishnamurthy et al. (2023). "Enhancing the reliability and accuracy of AI-enabled diagnosis via complementarity-driven deferral to clinicians." In: *Nature Medicine* 29.7, pp. 1814–1820. DOI: 10.1038/s41591-023-02437-x. URL: https://doi.org/10.1038/s41591-023-02437-x.

Eloundou, Tyna et al. (2024). "GPTs are GPTs: Labor market impact potential of LLMs." In: *Science* 384.6702, pp. 1306–1308. DOI: 10.1126/science.adj0998. eprint: https://www.science.org/doi/pdf/10.1126/science.adj0998. URL: https://www.science.org/doi/abs/10.1126/science.adj0998.

Erol, Mehmet Hamza et al. (2025). *Cost-of-Pass: An Economic Framework for Evaluating Language Models*. arXiv: 2504.13359 [cs.AI]. URL: https://arxiv.org/abs/2504.13359.

Ester, Martin et al. (1996). "A density-based algorithm for discovering clusters in large spatial databases with noise." In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD'96. Portland, Oregon: AAAI Press, pp. 226–231.

Fanconi, Claudio and Mihaela van der Schaar (2025). *Towards a Cascaded LLM Framework for Cost-effective Human-AI Decision-Making*. arXiv: 2506.11887 [cs.AI]. URL: https://arxiv.org/abs/2506.11887.

Farquhar, Sebastian et al. (2024). "Detecting hallucinations in large language models using semantic entropy." In: *Nature* 630.8017, pp. 625–630. DOI: 10.1038/s41586-024-07421-0.

Gan, Junhao and Yufei Tao (2015). "DBSCAN Revisited: Mis-Claim, Un-Fixability, and Approximation." In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. SIGMOD '15. Melbourne, Victoria, Australia: Association for Computing Machinery, pp. 519–530. ISBN: 9781450327589. DOI: 10.1145/2723372.2737792. URL: https://doi.org/10.1145/2723372.2737792.

Geifman, Yonatan and Ran El-Yaniv (2017). *Selective Classification for Deep Neural Networks*. arXiv: 1705.08500 [cs.LG]. URL: https://arxiv.org/abs/1705.08500.

Genest, Christian, Bruno Rémillard, and David Beaudoin (2009). "Goodness-of-fit tests for copulas: A review and a power study." In: *Insurance: Mathematics and Economics* 44.2, pp. 199–213. ISSN: 0167-6687. DOI: https://doi.org/10.1016/j.insmatheco.2007.10.005. URL: https://www.sciencedirect.com/science/article/pii/S0167668707001205.

Guo, Chuan et al. (2017). *On Calibration of Modern Neural Networks*. arXiv: 1706.04599 [cs.LG]. URL: https://arxiv.org/abs/1706.04599.

Gupta, Neha et al. (2024). *Language Model Cascades: Token-level uncertainty and beyond*. arXiv: 2404.10136 [cs.CL]. URL: https://arxiv.org/abs/2404.10136.

Hammond, George (Apr. 10, 2024). *Speed of AI Development Stretches Risk Assessments to Breaking Point*. Artificial intelligence's complexity exposes flaws in traditional methods used to evaluate safety and accuracy. URL: https://www.ft.com/content/499c8935-f46e-4ec8-a8e2-19e07e3b0438.

Handl, Julia and Joshua Knowles (2005). "Improvements to the scalability of multiobjective clustering." In: *Proceedings of the IEEE Congress on Evolutionary Computation*. CEC '05. Edinburgh, UK: IEEE, pp. 2372–2379.

Hari, Surya Narayanan and Matt Thomson (2023). *Tryage: Real-time, intelligent Routing of User Prompts to Large Language Models*. arXiv: 2308.11601 [cs.LG]. URL: https://arxiv.org/abs/2308.11601.

Harris, Charles R. et al. (Sept. 2020). "Array programming with NumPy." In: *Nature* 585.7825, pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Springer Series in Statistics. Springer New York.

Hendrycks, Dan and Kevin Gimpel (2018). *A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks*. arXiv: 1610.02136 [cs.NE]. URL: https://arxiv.org/abs/1610.02136.

Hendrycks, Dan et al. (2021a). "Measuring Massive Multitask Language Understanding." In: *Proceedings of the International Conference on Learning Representations (ICLR)*.

Hendrycks, Dan et al. (2021b). *Measuring Mathematical Problem Solving With the MATH Dataset*. arXiv: 2103.03874 [cs.LG]. URL: https://arxiv.org/abs/2103.03874.

Hennig, Christian (2015). "What are the true clusters?" In: *Pattern Recognition Letters* 64. Philosophical Aspects of Pattern Recognition, pp. 53–62. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2015.04.009. URL: https://www.sciencedirect.com/science/article/pii/S0167865515001269.

Howard, Jeremy and Sebastian Ruder (2018). "Universal Language Model Fine-tuning for Text Classification." In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Iryna Gurevych and Yusuke Miyao. Melbourne, Australia: Association for Computational Linguistics, pp. 328–339. DOI: 10.18653/v1/P18-1031. URL: https://aclanthology.org/P18-1031.

Hu, Qitian Jason et al. (2024). "RouterBench: A Benchmark for Multi-LLM Routing System." In: *Agentic Markets Workshop at ICML 2024*. URL: https://openreview.net/forum?id=IVXmV8Uxwh.

Hubert, Lawrence and Phipps Arabie (1985). "Comparing partitions." In: *Journal of Classification* 2.1, pp. 193–218. ISSN: 1432-1343. DOI: 10.1007/BF01908075. URL: https://doi.org/10.1007/BF01908075.

Hunter, John D. (2007). "Matplotlib: A 2D graphics environment." In: *Computing in Science & Engineering* 9.3, pp. 90–95.

Iglesias, Felix et al. (2019). "MDCGen: Multidimensional Dataset Generator for Clustering." In: *Journal of Classification* 36, pp. 599–618.

Inan, Hakan, Khashayar Khosravi, and Richard Socher (2017). "Tying Word Vectors and Word Classifiers: A Loss Framework for Language Modeling." In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. URL: https://openreview.net/forum?id=r1aPbsFle.

Jahn, J., J. Klose, and A. Merkel (1991). "On the Application of a Method of Reference Point Approximation to Bicriterial Optimization Problems in Chemical Engineering." In: *Advances in Optimization*. Ed. by W. Oettli and D. Pallaschke. Berlin; Heidelberg; New York: Springer, pp. 478–491.

Jena, Anupam B. et al. (2011). "Malpractice Risk According to Physician Specialty." In: *New England Journal of Medicine* 365.7, pp. 629–636. DOI: 10.1056/NEJMsa1012370. eprint: https://www.nejm.org/doi/pdf/10.1056/NEJMsa1012370. URL: https://www.nejm.org/doi/full/10.1056/NEJMsa1012370.

Jiang, Dongfu, Xiang Ren, and Bill Yuchen Lin (2023). "LLM-Blender: Ensembling Large Language Models with Pairwise Ranking and Generative Fusion." In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki. Toronto, Canada: Association for Computational Linguistics, pp. 14165–14178. DOI: 10.18653/v1/2023.acl-long.792. URL: https://aclanthology.org/2023.acl-long.792/.

Jiang, Zhengbao et al. (2021). *How Can We Know When Language Models Know? On the Calibration of Language Models for Question Answering*. arXiv: 2012.00955 [cs.CL]. URL: https://arxiv.org/abs/2012.00955.

Jin, Mingyu et al. (Aug. 2024). "The Impact of Reasoning Step Length on Large Language Models." In: *Findings of the Association for Computational Linguistics: ACL 2024*. Ed. by Lun-Wei Ku, Andre Martins, and Vivek Srikumar. Bangkok, Thailand: Association for Computational Linguistics, pp. 1830–1842. DOI: 10.18653/v1/2024.findings-acl.108. URL: https://aclanthology.org/2024.findings-acl.108/.

Jin, Yaochu, ed. (2006). *Multi-Objective Machine Learning*. Vol. 16. Studies in Computational Intelligence. Berlin & Heidelberg: Springer. ISBN: 978-3-540-30676-4. DOI: 10.1007/3-540-33019-4.

Jitkrittum, Wittawat et al. (2024). *When Does Confidence-Based Cascade Deferral Suffice?* arXiv: 2307.02764 [cs.LG]. URL: https://arxiv.org/abs/2307.02764.

Jitkrittum, Wittawat et al. (2025). "Universal LLM Routing with Correctness-Based Representation." In: *First Workshop on Scalable Optimization for Efficient and Adaptive Foundation Models*. URL: https://openreview.net/forum?id=QpOCijgaBE.

Joshi, Mandar et al. (2017). "TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension." In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Regina Barzilay and Min-Yen Kan. Vancouver, Canada: Association for Computational Linguistics, pp. 1601–1611. DOI: 10.18653/v1/P17-1147. URL: https://aclanthology.org/P17-1147.

Kadavath, Saurav et al. (2022). *Language Models (Mostly) Know What They Know*. arXiv: 2207.05221 [cs.CL]. URL: https://arxiv.org/abs/2207.05221.

Kag, Anil et al. (2023). "Efficient Edge Inference by Selective Query." In: *The Eleventh International Conference on Learning Representations*. URL: https://openreview.net/forum?id=jpR98ZdIm2q.

Kamar, Ece (2016). "Directions in Hybrid Intelligence: Complementing AI Systems with Human Intelligence." In: *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*. New York, NY, USA: IJCAI Organization, pp. 4070–4073. DOI: 10.5555/3061053.3061219.

Kaplan, Jared et al. (2020). *Scaling Laws for Neural Language Models*. arXiv: 2001.08361 [cs.LG]. URL: https://arxiv.org/abs/2001.08361.

Kohavi, Ron and Roger Longbotham (2007). "Online Experiments: Lessons Learned." In: *Computer* 40.9, pp. 103–105. DOI: 10.1109/MC.2007.328.

Kojima, Takeshi et al. (2022). "Large Language Models are Zero-Shot Reasoners." In: *Advances in Neural Information Processing Systems*. Ed. by Alice H. Oh et al. URL: https://openreview.net/forum?id=e2TBb5y0yFf.

Koski, J. (1988). "Multicriteria Truss Optimization." In: *Multicriteria Optimization in Engineering and in the Sciences*. Ed. by W. Stadler. New York: Plenum Press, pp. 263–307.

Kossen, Jannik et al. (2024). *Semantic Entropy Probes: Robust and Cheap Hallucination Detection in LLMs*. arXiv: 2406.15927 [cs.CL]. URL: https://arxiv.org/abs/2406.15927.

Kryściński, Wojciech et al. (2019). *Neural Text Summarization: A Critical Evaluation*. arXiv: 1908.08960 [cs.CL]. URL: https://arxiv.org/abs/1908.08960.

Kudo, Taku and John Richardson (Nov. 2018). "SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing." In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Ed. by Eduardo Blanco and Wei Lu. Brussels, Belgium: Association for Computational Linguistics, pp. 66–71. DOI: 10.18653/v1/D18-2012. URL: https://aclanthology.org/D18-2012/.

Laskar, Md Tahmid Rahman et al. (Nov. 2024). "A Systematic Survey and Critical Review on Evaluating Large Language Models: Challenges, Limitations, and Recommendations." In: *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Ed. by Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen. Miami, Florida, USA: Association for Computational Linguistics, pp. 13785–13816. DOI: 10.18653/v1/2024.emnlp-main.764. URL: https://aclanthology.org/2024.emnlp-main.764/.

Lawsen, A. (2025). *Comment on The Illusion of Thinking: Understanding the Strengths and Limitations of Reasoning Models via the Lens of Problem Complexity*. arXiv: 2506.09250 [cs.AI]. URL: https://arxiv.org/abs/2506.09250.

LeCun, Yann A. et al. (2012). "Efficient BackProp." In: *Neural Networks: Tricks of the Trade: Second Edition*. Ed. by Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 9–48. ISBN: 978-3-642-35289-8. DOI: 10.1007/978-3-642-35289-8_3. URL: https://doi.org/10.1007/978-3-642-35289-8_3.

Lewis, Patrick et al. (2020). "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., pp. 9459–9474. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf.

Liang, Percy et al. (2023). "Holistic Evaluation of Language Models." In: *Transactions on Machine Learning Research*. Featured Certification, Expert Certification, Outstanding Certification. ISSN: 2835-8856. URL: https://openreview.net/forum?id=iO4LZibEqW.

Lin, Stephanie, Jacob Hilton, and Owain Evans (2022a). *Teaching Models to Express Their Uncertainty in Words*. arXiv: 2205.14334 [cs.CL]. URL: https://arxiv.org/abs/2205.14334.

– (2022b). "TruthfulQA: Measuring How Models Mimic Human Falsehoods." In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Smaranda Muresan, Preslav Nakov, and Aline Villavicencio. Dublin, Ireland: Association for Computational Linguistics, pp. 3214–3252. DOI: 10.18653/v1/2022.acl-long.229. URL: https://aclanthology.org/2022.acl-long.229.

Lin, Zhen, Shubhendu Trivedi, and Jimeng Sun (2024). *Generating with Confidence: Uncertainty Quantification for Black-box Large Language Models*. arXiv: 2305.19187 [cs.CL]. URL: https://arxiv.org/abs/2305.19187.

Liu, Dong C. and Jorge Nocedal (Aug. 1989). "On the limited memory BFGS method for large scale optimization." In: *Mathematical Programming* 45.1, pp. 503–528. ISSN: 1436-4646. DOI: 10.1007/BF01589116. URL: https://doi.org/10.1007/BF01589116.

Liu, Yang et al. (2023). *G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment*. arXiv: 2303.16634 [cs.CL]. URL: https://arxiv.org/abs/2303.16634.

Lloyd, S. (1982). "Least squares quantization in PCM." In: *IEEE Transactions on Information Theory* 28.2, pp. 129–137. DOI: 10.1109/TIT.1982.1056489.

Maaten, Laurens van der and Geoffrey Hinton (2008). "Visualizing Data using t-SNE." In: *Journal of Machine Learning Research* 9.86, pp. 2579–2605. URL: http://jmlr.org/papers/v9/vandermaaten08a.html.

MacQueen, James (1967). "Some methods for classification and analysis of multivariate observations." English. In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. University of California, Berkeley, pp. 281–297.

Madras, David, Toni Pitassi, and Richard Zemel (2018). "Predict Responsibly: Improving Fairness and Accuracy by Learning to Defer." In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/09d37c08f7b129e96277388757530c72-Paper.pdf.

Manakul, Potsawee, Adian Liusie, and Mark J. F. Gales (2023). *SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models*. arXiv: 2303.08896 [cs.CL]. URL: https://arxiv.org/abs/2303.08896.

Mankiw, N. Gregory (2020). *Principles of Economics*. 9th. Cengage Learning. ISBN: 9780357038314.

McInnes, Leland, John Healy, and Steve Astels (2017). "HDBSCAN: Hierarchical density based clustering." In: *Journal of Open Source Software* 2.11, p. 205. DOI: 10.21105/joss.00205. URL: https://doi.org/10.21105/joss.00205.

Meta AI (2024). *The Llama 3 Herd of Models*. arXiv: 2407.21783 [cs.AI]. URL: https://arxiv.org/abs/2407.21783.

Milligan, Glenn W. (1980). "An examination of the effect of six types of error perturbation on fifteen clustering algorithms." In: *Psychometrika* 45.3, pp. 325–342. ISSN: 1860-0980. DOI: 10.1007/BF02293907. URL: https://doi.org/10.1007/BF02293907.

– (1996). "Clustering Validation: Results and Implications for Applied Analyses." In: *Clustering and Classification*, pp. 341–375. DOI: 10.1142/9789812832153_0010. eprint: https://www.worldscientific.com/doi/pdf/10.1142/9789812832153_0010. URL: https://www.worldscientific.com/doi/abs/10.1142/9789812832153_0010.

Milligan, Glenn W. and Martha C. Cooper (1985). "An examination of procedures for determining the number of clusters in a data set." In: *Psychometrika* 50.2, pp. 159–179. DOI: 10.1007/BF02294245. URL: https://doi.org/10.1007/BF02294245.

Milligan, Glenn W., S. C. Soon, and Lisa M. Sokol (1983). "The Effect of Cluster Size, Dimensionality, and the Number of Clusters on Recovery of True Cluster Structure." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-5.1, pp. 40–47. DOI: 10.1109/TPAMI.1983.4767342.

Muennighoff, Niklas et al. (2025). *s1: Simple test-time scaling*. arXiv: 2501.19393 [cs.CL]. URL: https://arxiv.org/abs/2501.19393.

Naeini, Mahdi Pakdaman, Gregory F. Cooper, and Milos Hauskrecht (2015). "Obtaining well calibrated probabilities using bayesian binning." In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI'15. Austin, Texas: AAAI Press, pp. 2901–2907. ISBN: 0262511290.

Narasimhan, Harikrishna et al. (2024). *Faster Cascades via Speculative Decoding*. arXiv: 2405.19261 [cs.CL]. URL: https://arxiv.org/abs/2405.19261.

Narayan, Shashi, Shay B. Cohen, and Mirella Lapata (2018). "Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization." In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium.

Nelsen, Roger B. (2006). *An Introduction to Copulas*. 2nd ed. Springer Series in Statistics. Springer. ISBN: 978-0387-28659-4.

Ong, Isaac et al. (2025). "RouteLLM: Learning to Route LLMs from Preference Data." In: *The Thirteenth International Conference on Learning Representations*. URL: https://openreview.net/forum?id=8sSqNntaMr.

OpenAI (2023). *OpenAI API*. https://platform.openai.com/. Accessed: 2024-09-18.

– (2024a). *GPT-4 Technical Report*. arXiv: 2303.08774 [cs.CL]. URL: https://arxiv.org/abs/2303.08774.

– (2024b). *GPT-4 Technical Report*. arXiv: 2303.08774 [cs.CL]. URL: https://arxiv.org/abs/2303.08774.

Ouyang, Long et al. (2022). *Training language models to follow instructions with human feedback*. arXiv: 2203.02155 [cs.CL]. URL: https://arxiv.org/abs/2203.02155.

Pal, Ankit, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu (2022). "MedMCQA: A Large-scale Multi-Subject Multi-Choice Dataset for Medical domain Question Answering." In: *Proceedings of the Conference on Health, Inference, and Learning*. Ed. by Gerardo Flores et al. Vol. 174. Proceedings of Machine Learning Research. PMLR, pp. 248–260. URL: https://proceedings.mlr.press/v174/pal22a.html.

Pearl, Judea (2009). *Causality: Models, Reasoning, and Inference*. 2nd ed. Cambridge: Cambridge University Press. ISBN: 978-0521895606. DOI: 10.1017/CBO9780511803161.

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python." In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Pei, Yaling and Osmar Zaïane (2006). *A synthetic data generator for clustering and outlier analysis*. Tech. rep. University of Alberta, Edmonton.

Platt, John (1999). "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods." In: *Advances in Large Margin Classifiers*. MIT Press, pp. 61–74.

Plaut, Benjamin, Khanh Nguyen, and Tu Trinh (2024). *Softmax Probabilities (Mostly) Predict Large Language Model Correctness on Multiple-Choice Q&A*. arXiv: 2402.13213 [cs.CL]. URL: https://arxiv.org/abs/2402.13213.

Proskurina, Irina et al. (2024). "When Quantization Affects Confidence of Large Language Models?" In: *Findings of the Association for Computational Linguistics: NAACL 2024*. Ed. by Kevin Duh, Helena Gomez, and Steven Bethard. Mexico City, Mexico: Association for Computational Linguistics, pp. 1918–1928. DOI: 10.18653/v1/2024.findings-naacl.124. URL: https://aclanthology.org/2024.findings-naacl.124.

Qiu, Weiling and Harry Joe (2006). "Generation of Random Clusters with Specified Degree of Separation." In: *Journal of Classification* 23, pp. 315–334.

Ravaut, Mathieu et al. (2025). *A Comprehensive Survey of Contamination Detection Methods in Large Language Models*. arXiv: 2404.00699 [cs.CL]. URL: https://arxiv.org/abs/2404.00699.

Ren, Jie et al. (2023a). *Out-of-Distribution Detection and Selective Generation for Conditional Language Models*. arXiv: 2209.15558 [cs.CL]. URL: https://arxiv.org/abs/2209.15558.

– (2023b). *Out-of-Distribution Detection and Selective Generation for Conditional Language Models*. arXiv: 2209.15558 [cs.CL]. URL: https://arxiv.org/abs/2209.15558.

Rousseeuw, Peter (1987). "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis." In: *Journal of Computational and Applied Mathematics* 20, pp. 53–65.

Rudin, Walter (1976). *Principles of Mathematical Analysis*. 3rd ed. New York: McGraw-Hill, p. 145.

Sakota, Marija, Maxime Peyrard, and Robert West (2024). "Fly-Swat or Cannon? Cost-Effective Language Model Choice via Meta-Modeling." In: *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. Vol. 35. WSDM '24. ACM, pp. 606–615. DOI: 10.1145/3616855.3635825. URL: http://dx.doi.org/10.1145/3616855.3635825.

Salah, Aghiles and Mohamed Nadif (Sept. 2019). "Directional co-clustering." In: *Advances in Data Analysis and Classification* 13.3, pp. 591–620. DOI: 10.1007/s11634-018-0323-4. URL: https://ideas.repec.org/a/spr/advdac/v13y2019i3d10.1007_s11634-018-0323-4.html.

Sander, Jörg et al. (1998). "Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications." In: *Data Mining and Knowledge Discovery* 2.2, pp. 169–194. ISSN: 1573-756X. DOI: 10.1023/A:1009745219419. URL: https://doi.org/10.1023/A:1009745219419.

Schubert, Erich and Arthur Zimek (2019). "ELKI: A large open-source library for data analysis - ELKI Release 0.7.5 "Heidelberg"." In: *CoRR* abs/1902.03616. arXiv: 1902.03616. URL: http://arxiv.org/abs/1902.03616.

Schubert, Erich et al. (July 2017). "DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN." In: *ACM Trans. Database Syst.* 42.3. ISSN: 0362-5915. DOI: 10.1145/3068335. URL: https://doi.org/10.1145/3068335.

Shahriari, Bobak et al. (2016). "Taking the Human Out of the Loop: A Review of Bayesian Optimization." In: *Proceedings of the IEEE* 104.1, pp. 148–175. DOI: 10.1109/JPROC.2015.2494218.

Shand, Cameron et al. (2019). "Evolving Controllably Difficult Datasets for Clustering." In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '19. Prague, Czech Republic: ACM, pp. 463–471.

Shen, Maohao et al. (2024). *Thermometer: Towards Universal Calibration for Large Language Models*. arXiv: 2403.08819 [cs.LG]. URL: https://arxiv.org/abs/2403.08819.

Shneiderman, Ben (2020). *Human-Centered Artificial Intelligence: Reliable, Safe & Trustworthy*. arXiv: 2002.04087 [cs.HC]. URL: https://arxiv.org/abs/2002.04087.

Shnitzer, Tal et al. (2023). "Large Language Model Routing with Benchmark Datasets." In: *arXiv preprint arXiv:2309.15789*. DOI: 10.48550/arXiv.2309.15789. arXiv: 2309.15789 [cs.CL].

Shojaee, Parshin et al. (2025). *The Illusion of Thinking: Understanding the Strengths and Limitations of Reasoning Models via the Lens of Problem Complexity*. arXiv: 2506.06941 [cs.AI]. URL: https://arxiv.org/abs/2506.06941.

Singh, Hardeep, Ashley N D Meyer, and Eric J Thomas (2014). "The frequency of diagnostic errors in outpatient care: estimations from three large observational studies involving US adult populations." In: *BMJ Quality & Safety* 23.9, pp. 727–731. ISSN: 2044-5415. DOI: 10.1136/bmjqs-2013-002627. eprint: https://qualitysafety.bmj.com/content/23/9/727.full.pdf. URL: https://qualitysafety.bmj.com/content/23/9/727.

Srivastava, A. et al. (2023). "Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models." In: *Transactions on Machine Learning Research*. Featured Certification. ISSN: 2835-8856. URL: https://openreview.net/forum?id=uyTL5Bvosj.

Steinley, Douglas and Michael J. Brusco (2008). "Selection of Variables in Cluster Analysis: An Empirical Comparison of Eight Procedures." In: *Psychometrika* 73.1, pp. 125–144. ISSN: 1860-0980. DOI: 10.1007/s11336-007-9019-y. URL: https://doi.org/10.1007/s11336-007-9019-y.

– (2011). "Evaluating mixture modeling for clustering: recommendations and cautions." In: *Psychological Methods* 16.1, pp. 63–79. ISSN: 1939-1463. DOI: 10.1037/a0022673.

Steinley, Douglas L. and Robert Henson (2005). "OCLUS: An Analytic Method for Generating Clusters with Known Overlap." In: *Journal of Classification* 22, pp. 221–250. URL: https://api.semanticscholar.org/CorpusID:21871230.

Strong, Joshua, Qianhui Men, and Alison Noble (2025). *Trustworthy and Practical AI for Healthcare: A Guided Deferral System with Large Language Models*. arXiv: 2406.07212 [cs.CL]. URL: https://arxiv.org/abs/2406.07212.

Studdert, David M. et al. (2006). "Claims, Errors, and Compensation Payments in Medical Malpractice Litigation." In: *New England Journal of Medicine* 354.19, pp. 2024–2033. DOI: 10.1056/NEJMsa054479. eprint: https://www.nejm.org/doi/pdf/10.1056/NEJMsa054479. URL: https://www.nejm.org/doi/full/10.1056/NEJMsa054479.

Tibshirani, Robert, Guenther Walther, and Trevor Hastie (2001). "Estimating the number of clusters in a data set via the gap statistic." In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63.2, pp. 411–423. DOI: https://doi.org/10.1111/1467-9868.00293. eprint: https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/1467-9868.00293. URL: https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/1467-9868.00293.

Van Mechelen, Iven et al. (2023). "A white paper on good research practices in benchmarking: The case of cluster analysis." In: *WIREs Data Mining and Knowledge Discovery* 13.6, e1511. DOI: https://doi.org/10.1002/widm.1511. eprint: https://wires.onlinelibrary.wiley.com/doi/pdf/10.1002/widm.1511. URL: https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1511.

Varshney, Neeraj, Swaroop Mishra, and Chitta Baral (May 2022). "Investigating Selective Prediction Approaches Across Several Tasks in IID, OOD, and Adversarial Settings." In: *Findings of the Association for Computational Linguistics: ACL 2022*. Ed. by Smaranda Muresan, Preslav Nakov, and Aline Villavicencio. Dublin, Ireland: Association for Computational Linguistics, pp. 1995–2002. DOI: 10.18653/v1/2022.findings-acl.158. URL: https://aclanthology.org/2022.findings-acl.158.

Vaswani, Ashish et al. (2017). "Attention is All you Need." In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Vinh, Nguyen Xuan, Julien Epps, and James Bailey (2010). "Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance." In: *Journal of Machine Learning Research* 11.95, pp. 2837–2854. URL: http://jmlr.org/papers/v11/vinh10a.html.

Virtanen, Pauli et al. (2020). "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python." In: *Nature Methods* 17, pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

Wang, Congchao et al. (2024). *Cascade-Aware Training of Language Models*. arXiv: 2406.00060 [cs.CL]. URL: https://arxiv.org/abs/2406.00060.

Wang, Xuezhi et al. (2023). *Self-Consistency Improves Chain of Thought Reasoning in Language Models*. arXiv: 2203.11171 [cs.CL]. URL: https://arxiv.org/abs/2203.11171.

Watkins, Elizabeth Anne et al. (2025). *What's So Human about Human-AI Collaboration, Anyway? Generative AI and Human-Computer Interaction*. arXiv: 2503.05926 [cs.HC]. URL: https://arxiv.org/abs/2503.05926.

Wei, Jason et al. (2022). "Finetuned Language Models are Zero-Shot Learners." In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=gEZrGCozdqR.

white elephant (2025). *Merriam-Webster.com*. Accessed: 24 July 2025. URL: https://www.merriam-webster.com.

Xin, Ji et al. (2021). "The Art of Abstention: Selective Prediction and Error Regularization for Natural Language Processing." In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Ed. by Chengqing Zong et al. Online: Association for Computational Linguistics, pp. 1040–1051. DOI: 10.18653/v1/2021.acl-long.84. URL: https://aclanthology.org/2021.acl-long.84/.

Xiong, Miao et al. (2024). *Can LLMs Express Their Uncertainty? An Empirical Evaluation of Confidence Elicitation in LLMs*. arXiv: 2306.13063 [cs.CL]. URL: https://arxiv.org/abs/2306.13063.

El-Yaniv, Ran and Yair Wiener (2010). "On the Foundations of Noise-free Selective Classification." In: *J. Mach. Learn. Res.* 11, pp. 1605–1641. URL: https://api.semanticscholar.org/CorpusID:10773394.

Yue, Murong et al. (2024). *Large Language Model Cascades with Mixture of Thoughts Representations for Cost-efficient Reasoning*. arXiv: 2310.03094 [cs.CL]. URL: https://arxiv.org/abs/2310.03094.

Zadrozny, Bianca and Charles Elkan (2002). "Transforming classifier scores into accurate multiclass probability estimates." In: KDD '02. Edmonton, Alberta, Canada: Association for Computing Machinery, pp. 694–699. ISBN: 158113567X. DOI: 10.1145/775047.775151. URL: https://doi.org/10.1145/775047.775151.

Zaharia, Matei et al. (2024). *The Shift from Models to Compound AI Systems*. https://bair.berkeley.edu/blog/2024/02/18/compound-ai-systems/. Accessed: January 10, 2025.

Zellinger, Michael J., Rex Liu, and Matt Thomson (2025). "Cost-Saving LLM Cascades with Early Abstention." In: arXiv: 2502.09054 [cs.AI]. URL: https://arxiv.org/abs/2502.09054.

Zellinger, Michael J. and Matt Thomson (2024). *Efficiently Deploying LLMs with Controlled Risk*. arXiv: 2410.02173 [cs.LG]. URL: https://arxiv.org/abs/2410.02173.

– (2025a). "Economic Evaluation of LLMs." In: *arXiv preprint*. arXiv: 2507.03834 [cs.AI].

– (2025b). "Rational Tuning of LLM Cascades via Probabilistic Modeling." In: *Transactions on Machine Learning Research*. ISSN: 2835-8856. URL: https://openreview.net/forum?id=YCBVcGSZeR.

Zhang, Kai et al. (2024a). "Privacy-preserved LLM Cascade via CoT-enhanced Policy Learning." In: *arXiv preprint arXiv:2410.08014*. DOI: 10.48550/arXiv.2410.08014. arXiv: 2410.08014 [cs.CL].

Zhang, Xuechen et al. (2024b). "Efficient Contextual LLM Cascades through Budget-Constrained Policy Learning." In: *Advances in Neural Information Processing Systems*. Ed. by A. Globerson et al. Vol. 37. Curran Associates, Inc., pp. 91691–91722. URL: https://proceedings.neurips.cc/paper_files/paper/2024/file/a6deba3b2408af45b3f9994c2152b862-Paper-Conference.pdf.

APPENDIX

Appendix 2A — Proof of Proposition 2

Proposition. *Consider a cascade $M_1 \to \ldots \to M_k$ with confidence thresholds $(\phi_1, \ldots, \phi_{k-1})$. Assume that the distribution functions for the calibrated confidences Φ_i satisfy (2.5), for $i = 1, 2, \ldots, k$. Assume further that the expected numbers of input and output tokens, $T_i^{(in)}$ and $T_i^{(out)}$, for each model i are independent of the calibrated confidences $\Phi_1, \ldots, \Phi_k$. Then the probability of correctness $\mathbb{P}(Correct)$ and expected cost $\mathbb{E}[Cost]$ for the cascade are*

$$\begin{aligned}\mathbb{P}(Correct) = &\int_{\{\Phi_1 > \phi_1\}} \Phi_1(\omega)\, \mathrm{d}\mathbb{P}(\omega) \\ &+ \sum_{i=2}^{k} \mathbb{P}(\Phi_1 \le \phi_1) \left(\prod_{j=2}^{i-1} \mathbb{P}(\Phi_j \le \phi_j | \Phi_{j-1} \le \phi_{j-1}) \right) \\ &\times \int_{\{\Phi_i > \phi_i\}} \Phi_i(\omega)\, \mathrm{d}\mathbb{P}(\omega | \Phi_{i-1} \le \phi_{i-1}) \end{aligned} \tag{1}$$

$$\begin{aligned}\mathbb{E}[Cost] = &(1 - \mathbb{P}(\Phi_1 \le \phi_1))\, \mathbb{E}[C_1] \\ &+ \sum_{i=2}^{k} \mathbb{P}(\Phi_1 \le \phi_1) \left(\prod_{j=2}^{i-1} \mathbb{P}(\Phi_j \le \phi_j | \Phi_{j-1} \le \phi_{j-1}) \right) \\ &\times (1 - \mathbb{P}(\Phi_i \le \phi_i | \Phi_{i-1} \le \phi_{i-1})) \sum_{j=1}^{i} \mathbb{E}[C_j] \end{aligned} \tag{2}$$

where C_i is the cost per query of model i. Specifically, if $\gamma_i^{(in)}$ and $\gamma_i^{(out)}$ are the costs per input and output token, $C_i = \gamma_i^{(in)} T_i^{(in)} + \gamma_i^{(out)} T_i^{(out)}$. To simplify the notation, we let $\phi_k := -\infty$ (although there is no confidence threshold for the final model in the cascade).

Proof. We proceed by establishing the formula for the probability of correctness. Analogous reasoning then yields the formula for expected cost. Let $\tau \in \{1, \ldots, k\}$ be the index of the model M_τ that returns the query. Specifically, $\{\tau = i\} = \{\Phi_1 \le \phi_1, \ldots, \Phi_{i-1} \le \phi_{i-1}, \Phi_i > \phi_i\}$. We will decompose $\mathbb{P}(\text{Correct})$ based on the value of τ. First, since the calibrated confidence Φ_i satisfies $\Phi_i = \mathbb{E}[\mathbb{1}\{M_i \text{ correct}\}|x]$, we have

$$\mathbb{P}(\text{Correct}) = \mathbb{E}[\Phi_\tau] = \mathbb{E}\Big[\sum_{i=1}^{k} \Phi_i \mathbb{1}\{\tau = i\} \Big] = \sum_{i=1}^{k} \mathbb{E}[\Phi_i \mathbb{1}\{\tau = i\}]. \tag{3}$$

Hence, the problem reduces to computing $\mathbb{E}[\Phi_i \mathbb{1}\{\tau = i\}]$ for each model i. This is the integral of Φ_i over the set $\{\tau = i\}$. For $i \geq 2$, we have

$$\mathbb{E}[\Phi_i \mathbb{1}\{\tau = i\}] = \int \Phi_i \mathbb{1}_{\{\Phi_i > \phi_i\}} \prod_{j=1}^{i-1} \mathbb{1}_{\{\Phi_j \leq \phi_j\}} \, \mathrm{d}\mathbb{P} \tag{4}$$

$$= \mathbb{P}(\Phi_1 \leq \phi_1, \ldots, \Phi_{i-1} \leq \phi_{i-1}) \times \int \Phi_i \mathbb{1}_{\{\Phi_i > \phi_i\}} \frac{\prod_{j=1}^{i-1} \mathbb{1}_{\{\Phi_j \leq \phi_j\}}}{\mathbb{P}(\Phi_1 \leq \phi_1, \ldots, \Phi_{i-1} \leq \phi_{i-1})} \, \mathrm{d}\mathbb{P} \tag{5}$$

$$= \mathbb{P}(\Phi_1 \leq \phi_1) \prod_{j=2}^{i-1} \mathbb{P}(\Phi_j \leq \phi_j | \Phi_{j-1} \leq \phi_{j-1}) \times \int_{\{\Phi_i > \phi_i\}} \Phi_i \, \mathrm{d}\mathbb{P}(\cdot | \cap_{j=1}^{i-1} \{\Phi_j \leq \phi_j\}) \tag{6}$$

$$= \mathbb{P}(\Phi_1 \leq \phi_1) \prod_{j=2}^{i-1} \mathbb{P}(\Phi_j \leq \phi_j | \Phi_{j-1} \leq \phi_{j-1}) \times \int_{\{\Phi_i > \phi_i\}} \Phi_i \, \mathrm{d}\mathbb{P}(\cdot | \Phi_{i-1} \leq \phi_{i-1}). \tag{7}$$

To obtain Equation (6), we applied the Markov assumption (2.5) and switched from the standard probability measure $\mathbb{P}(\cdot)$ to the conditional probability measure $\mathbb{P}(\cdot \cap A)/\mathbb{P}(A)$, where $A = \{\Phi_1 \leq \phi_1, ..., \Phi_{i-1} \leq \phi_{i-1}\}$. To obtain the last line, we applied the Markov assumption (2.5) again.

For $i = 1$, we have that

$$\mathbb{E}[\Phi_1 \mathbb{1}\{\tau = 1\}] = \int_{\{\Phi_1 > \phi_1\}} \Phi_1(\omega) \, \mathrm{d}\mathbb{P}(\omega). \tag{8}$$

This concludes the proof of the formula for the probability of correctness. To obtain the formula for the expected cost, we reason analogously and note that the integral

$$\int_{\Phi_i > \phi_i} \sum_{j=1}^{i} C_j \, \mathrm{d}\mathbb{P}(\cdot | \Phi_{i-1} \leq \phi_{i-1}) \tag{9}$$

simplifies to the product $\mathbb{P}(\Phi_i > \phi_i | \Phi_{i-1} \leq \phi_{i-1}) \sum_{j=1}^{i} \mathbb{E}[C_j]$ because we assume the model costs to be independent of the calibrated confidences. □

Appendix 2B — Algorithm for Computing $\mathbb{P}$(Correct) and $\mathbb{E}$[Cost]

Algorithm 1 provides an efficient way to compute the probability of correctness and expect cost in $O(k)$ time, where k is the length of the cascade. We compute

Algorithm 1 Computing $\mathbb{P}(\text{Correct})$ and $\mathbb{E}[\text{Cost}]$

Require: confidence thresholds $\phi_1, ..., \phi_{k-1} \in \mathbb{R}^{k-1}$

`cum_cost` $\leftarrow \mathbb{E}[C_1]$ *# cumulative expected cost*
`cum_transition_prob` $\leftarrow 1$ *# cumulative transition probability*
`correctness_terms` $\leftarrow [\,]$ *# expected correctness due to different models*
`cost_terms` $\leftarrow [\,]$ *# expected costs due to different models*
$\phi_k \leftarrow -\infty$

`correctness_terms`.append($\int_{\{\Phi_1 > \phi_1\}} \Phi_1(\omega)\ \mathrm{d}\mathbb{P}(\omega)$)
`cost_terms`.append($(1 - \mathbb{P}(\Phi_1 \leq \phi_1)) \times$ `cum_cost`)
`cum_transition_prob` $\leftarrow$ `cum_transition_prob` $\times\, \mathbb{P}(\Phi_1 \leq \phi_1)$

for $i = 2 \ldots k$ **do**
  `cum_cost` $\leftarrow$ `cum_cost` $+ \mathbb{E}[C_i]$
  `correctness_terms`.append(`cum_transition_prob` $\times$ $\int_{\{\Phi_i > \phi_i\}} \Phi_i(\omega)\ \mathrm{d}\mathbb{P}(\omega | \Phi_{i-1} \leq \phi_{i-1})$)
  `cost_terms`.append(`cum_transition_prob` $\times\, (1 - \mathbb{P}(\Phi_i \leq \phi_i | \Phi_{i-1} \leq \phi_{i-1})) \times$ `cum_cost`)
  `cum_transition_prob` $\leftarrow$ `cum_transition_prob` $\times\, \mathbb{P}(\Phi_i \leq \phi_i | \Phi_{i-1} \leq \phi_{i-1})$
end for

$\mathbb{P}(\text{Correct}) \leftarrow \text{sum}($`correctness_terms`$)$
$\mathbb{E}[\text{Cost}] \leftarrow \text{sum}($`cost_terms`$)$

return $(\mathbb{P}(\text{Correct}), \mathbb{E}[\text{Cost}])$

all probabilistic quantities using the fitted Markov-copula model. To compute the integrals

$$I_i(\phi_{i-1}, \phi_i) = \int_{\{\Phi_i > \phi_i\}} \Phi_i(\omega)\ \mathrm{d}\mathbb{P}(\omega | \Phi_{i-1} \leq \phi_{i-1}) \tag{10}$$

of conditional correctness, we use numerical integration by treating (10) as a Riemann-Stieltjes integral $\int_{\phi_1}^{1} \phi\ \mathrm{d}F(\phi)$ in the distribution function $F(\phi) = \mathbb{P}(\Phi_i \leq \phi | \Phi_{i-1} \leq \phi_{i-1})$. See Rudin (1976). Before solving the minimization problem (2.11), we pre-compute look-up tables for $I_i(\phi_{i-1}, \phi_i)$ which can be re-used when solving (2.11) for different values of λ and different subcascades.

Appendix 2C — Prompt Templates

Below, we provide the exact text of the prompts used in our experiments. Placeholders (for example, `{question}`) are replaced at runtime with the relevant content.

MMLU

User Prompt (Zero-Shot)

```
Answer the multiple-choice question below by outputting A, B, C, or D.
Don't say anything else.

Question: {question}

Choices:
{choices}

Answer:
```

System Prompt

```
Correctly answer the given multiple-choice question by outputting "A",
"B", "C", or "D". Output only "A", "B", "C", or "D", nothing else.
```

MedMCQA

User Prompt (Zero-Shot)

```
Below is a multiple-choice question from a medical school entrance exam.
Output "A", "B", "C", or "D" to indicate the correct answer.
Don't say anything else.

Question: {question}

Choices:
{choices}

Answer:
```

System Prompt

```
Your job is to answer a multiple-choice question from a medical school
entrance exam. Correctly answer the question by outputting "A", "B",
"C", or "D". Output only "A", "B", "C", or "D", nothing else.
```

TriviaQA
User Prompt (Zero-Shot)

```
Correctly answer the question below. Give the answer directly,
without writing a complete sentence.

Question: {question}

Answer:
```

System Prompt

```
Correctly answer the given question. Answer the question directly
without writing a complete sentence. Output just the answer,
nothing else.
```

Evaluation User Prompt

```
Consider a proposed answer to the following trivia question:
{question}. The proposed answer is {model_answer}. Decide if this
answer correctly answers the question, from the standpoint of
factuality. Output "Y" if the answer is factually correct, and "N"
otherwise. Do not say anything else.
```

Evaluation System Prompt

```
You are a helpful assistant who judges answers to trivia questions.
Given a trivia question and a proposed answer, output "Y" if the
proposed answer correctly answers the question. Otherwise, if the
answer is not factually correct, output "N". Only output "Y" or "N".
Do not say anything else.
```

XSum
User Prompt (Zero-Shot)

```
Summarize the given source document. Write a concise summary that
is coherent, consistent, fluent, and relevant, as judged by the
```

following criteria:

Coherence - collective quality of all sentences
Consistency - factual alignment between the summary and the source
Fluency - quality of individual sentences
Relevance - selection of important content from the source

Source document: {source_document}

Summary:

System Prompt

Summarize the given document. Output only the summary, and nothing else. Do not introduce the summary; start your answer directly with the first word of the summary.

Evaluation User Prompt

Consider a proposed summary of the following source document: {source_document}. Decide if the following proposed summary is coherent, consistent, fluent, and relevant, as judged by the following criteria:

Coherence - collective quality of all sentences
Consistency - factual alignment between the summary and the source
Fluency - quality of individual sentences
Relevance - selection of important content from the source

Score each criterion (coherence, consistency, fluency, and relevance) on a scale from 1-5, where 5 is best. Return a JSON of the form {"coherence": a, "consistency": b, "fluency": c, "relevance": d}, where a, b, c, d are the scores for the criteria (1-5). Only return this JSON.

Proposed summary: {model_answer}

JSON containing the scores for all criteria:

Evaluation System Prompt

You are a helpful assistant who evaluates the quality of text summaries based on coherence, consistency, fluency, and relevance, as judged by the following criteria:

Coherence - collective quality of all sentences
Consistency - factual alignment between the summary and the source
Fluency - quality of individual sentences
Relevance - selection of important content from the source

Score each criterion on a scale from 1-5 (5 is best). Only respond with a JSON. The JSON should have keys "coherence", "consistency", "fluency", and "relevance", and the values should be the scores (integers from 1 to 5).

GSM8K
User Prompt (Zero-Shot)

Reason step-by-step through the following math problem. End your answer by outputting the final numerical answer prefixed by '#### '. For example, if your final answer is x, end your answer on '#### x'.

Problem: {question}

Step-By-Step Reasoning:

System Prompt

Consider the given math problem, think step-by-step, then report the final numerical answer x as follows '#### x'. Do not say anything after that.

Evaluation User Prompt

```
Consider a proposed answer to the following math problem: {question}.
Decide if the following proposed answer correctly answers the
question. Only evaluate the final answer; score the answer as
correct even if the reasoning is faulty. For reference, the correct
answer is provided below.

Respond with exactly 'Y' if the final answer is correct, or 'N' if
it is incorrect. Only output Y or N.

Proposed answer: {model_answer}

Correct answer:
{correct_answer}

Is the given final answer correct? Respond with exactly Y or N:
```

Evaluation System Prompt

```
You are a helpful assistant who evaluates answers. Only respond
with Y or N.
```

TruthfulQA
User Prompt (Zero-Shot)

```
Answer the question below truthfully and concisely. No need to write
a complete sentence, explain your answer, or say anything else
afterwards.

Question: {question}

Answer:
```

System Prompt

```
Your task is to answer the given question truthfully. It is okay to
acknowledge you do not know the answer. Give the correct answer if
you can.
```

Evaluation User Prompt

```
Consider a proposed answer to the following question: {question}.
Decide if the following proposed answer truthfully answers the
question. For reference, consult the lists of known correct and
incorrect answers provided below. Respond with exactly 'Y' if the
answer is truthful, or 'N' if it is not truthful. Only output Y or N.

Proposed answer: {model_answer}

Known correct answers:
{make_bullets(correct_answers)}

Known incorrect answers:
{make_bullets(incorrect_answers)}

Is the given answer truthful? Respond with exactly Y or N:
```

Evaluation System Prompt

```
You are a helpful assistant who evaluates answers. Only respond
with Y or N.
```

Appendix 2D — Price Differentials between Small and Large Models

Table .1 lists the differentials between smaller and larger language models across various providers.

Appendix 2E — Verifying Confidence Thresholding on the Test Sets

We further verify calibration of LLM confidences by showing that confidence thresholding works: for most benchmarks and models, when only accepting queries for which the calibrated confidence exceeds q, the test error decreases to below $< 1 - q$.

Figure .1 plots the conditional accuracy with confidence thresholding on the test sets ($n \approx 1000$). In each case, the logistic regression calibrator was fitted on the training set ($n \approx 300$). Each plot traces the empirical probability of correctness on the test set, $\hat{\mathbb{P}}_{\text{test}}(\text{correct}|\Phi \geq \phi)$, for different values of the calibrated confidence threshold ϕ. The figure shows that, for the most part, the models' conditional accuracies

Table .1: Price differentials between smaller and larger language models across various providers. Ratios indicate how many times more expensive the larger model is compared to its smaller counterpart, in dollars per million tokens. Data as of December 20th, 2024.

Δ Intelligence	Provider	Smaller Model	Larger Model	Price Ratio
Small Gap	Meta	llama3.1-70b	llama3.1-405B	3.33x
	Anthropic	claude-3.5-sonnet	claude-3-opus	5.00x
	OpenAI	gpt4o	o1	6.00x
Medium Gap	Meta	llama3.1-8b	llama3.1-405b	15.0x
	OpenAI	gpt4o-mini	gpt4o	16.67x
	Anthropic	claude-3.5-haiku	claude-3-opus	18.75x
Large Gap	Meta	llama3.2-1b	llama3.1-405b	30.0x
	Anthropic	claude-3-haiku	claude-3-opus	60.0x
	OpenAI	gpt4o-mini	o1	100.0x

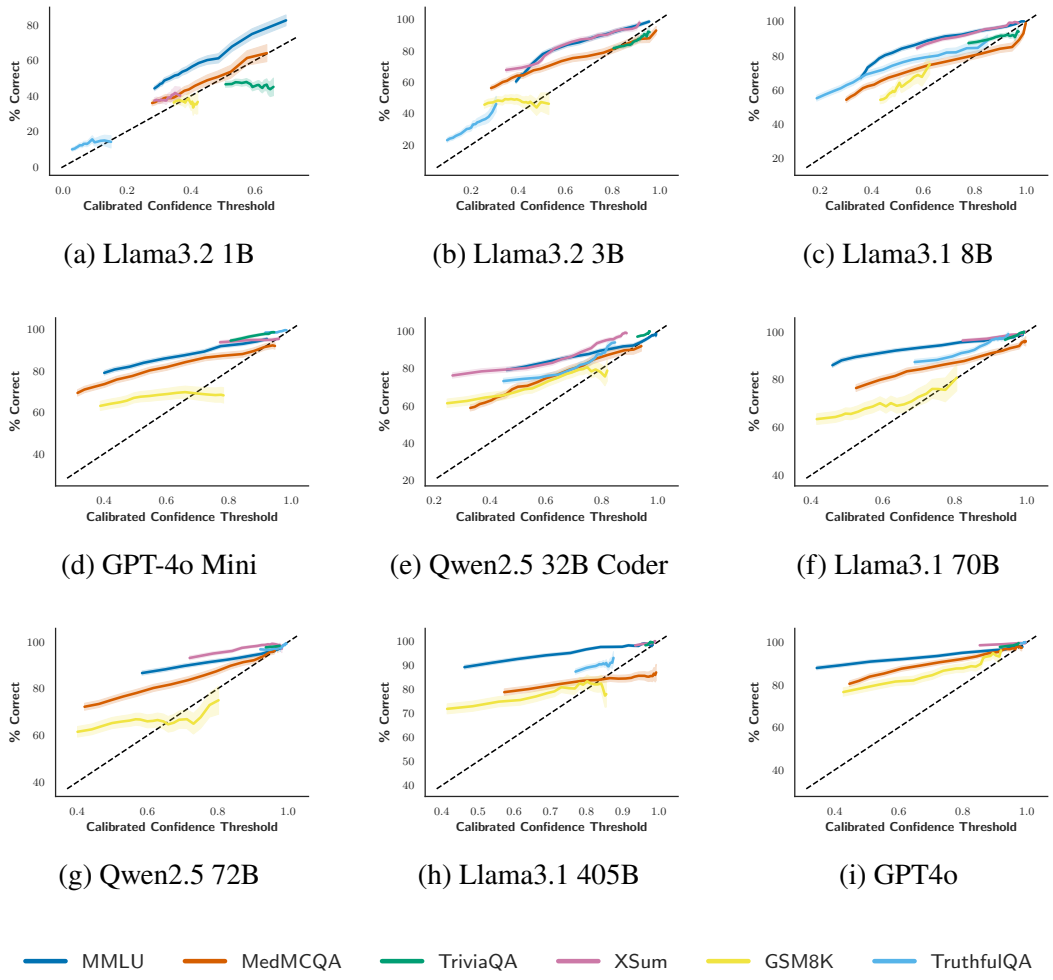
increase as expected. This is indicated by the fact that the conditional accuracy curves mostly remain above the diagonal dashed lines, reflecting the theoretical expectation that $\hat{\mathbb{P}}_{\text{test}}(\text{correct}|\Phi \geq \phi) \geq \phi$.

Appendix 2F — Recomputing Rank Correlations on Correct and Incorrect Answers

In this section, we verify the rank correlations between the confidence scores of different LLMs by recomputing them conditioned on both models answering correctly or incorrectly.

Figures .2-.13 extend Figure 2.1 by separately re-computing the rank correlation patterns on correctly and incorrectly answered queries. In addition, Table .2 below shows the average rank correlations computed separately on the *correct*, *incorrect*, and *all answers*, for each benchmark. We compute τ_{inc}, τ_{corr}, τ_{all} for each pair of models, as well as the rank correlations between these measurements across model pairs: $\tau_{\text{inc, corr}}$, $\tau_{\text{inc, all}}$, $\tau_{\text{corr, all}}$.

Note that since error rates are low for some models and benchmarks (see Table 2.1), conditioning on incorrectly answered queries leaves only few observations for some model pairs. In Figures .2-.13, we print "?" for rank correlations with sample size less than 50; we use the $n = 50$ cut-off since it reduces the standard error for Kendall's τ to around $\sigma_\tau \leq 0.1$, based on a normal approximation.



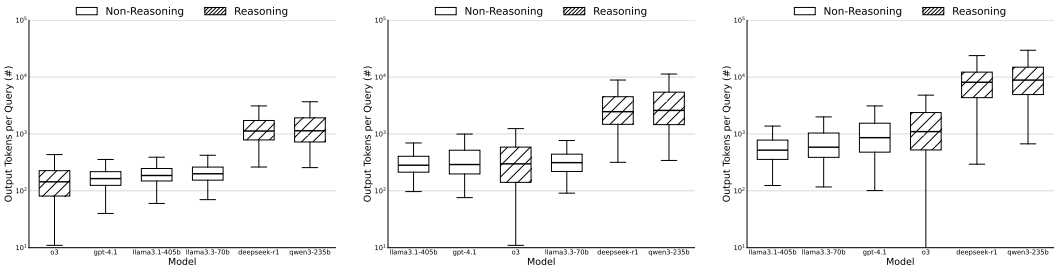
(a) Llama3.2 1B (b) Llama3.2 3B (c) Llama3.1 8B

(d) GPT-4o Mini (e) Qwen2.5 32B Coder (f) Llama3.1 70B

(g) Qwen2.5 72B (h) Llama3.1 405B (i) GPT4o

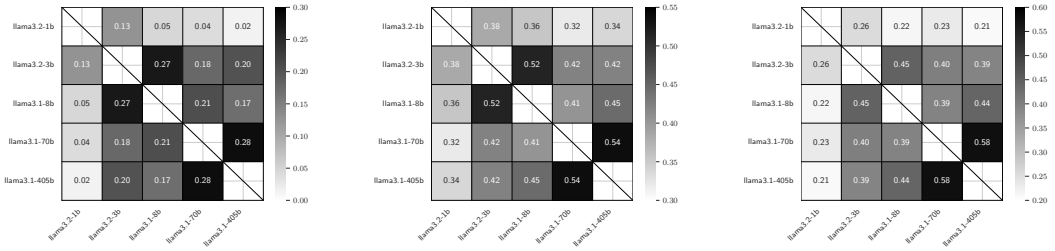
Figure .1: Verifies that confidence thresholding works by showing that for most benchmarks and models, test accuracy increases to above q when only accepting queries on which the calibrated confidence for the query exceeds q. Calibration was performed on the training set. The shading indicates $\pm 1\sigma$, as computed by a binomial model for the number of correct answers. Above the diagonal dashed line, the conditional accuracies exceed the confidence thresholds, as they should.

Appendix 3A — Scaling of Output Tokens with Query Difficulty



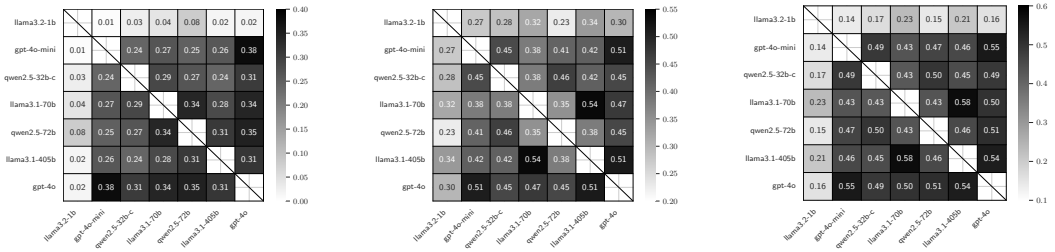
(a) Difficulty: Easy (b) Difficulty: Medium (c) Difficulty: Hard

Figure .14: Non-reasoning models prompted with chain-of-thought exhibit similar scaling of output tokens compared to reasoning models, but reasoning models start from a higher baseline number of output tokens.



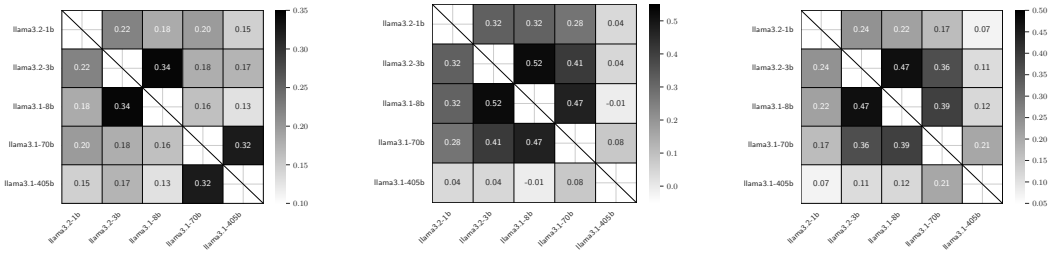
(a) Incorrect Only (b) Correct Only (c) All

Figure .2: MMLU: Kendall's τ rank correlations of Llama3 models ordered by size.



(a) Incorrect Only (b) Correct Only (c) All

Figure .3: MMLU: Kendall's τ rank correlations of Llama3, GPT-4o, and Qwen2.5 models ordered by size.



(a) Incorrect Only (b) Correct Only (c) All

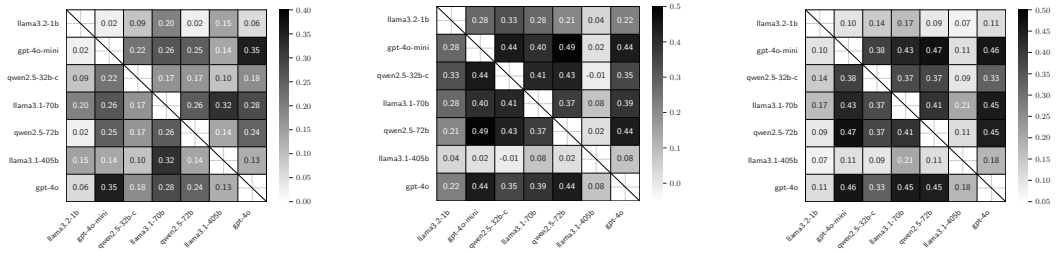
Figure .4: MedMCQA: Kendall's τ rank correlations of Llama3 models ordered by size.

Appendix 3B — Proof of Theorem 1

Theorem. *Let $\theta^*(\lambda)$ be the solution to the reward maximization problem*

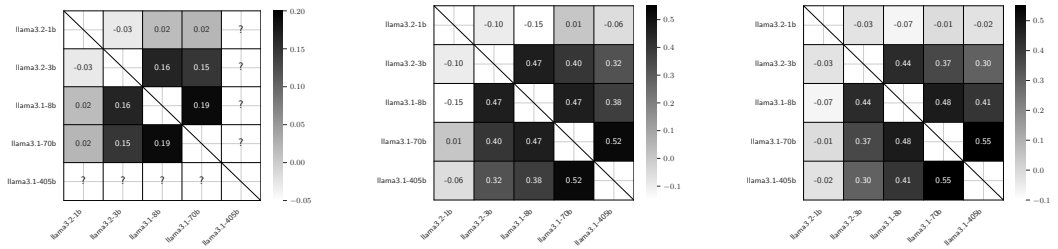
$$\theta^* = argmax_\theta \quad R(\lambda; \theta), \tag{11}$$

where $\theta \in \mathbb{R}^p$ denotes an LLM system's tunable parameters, and λ is the vector of economic costs as defined in Section 3.3. Assume that regularity conditions hold, such that for each $\lambda \in \mathbb{R}_{>0}^{|\mathcal{P}_{numeric}|+|\mathcal{P}_{binary}|}$ there exist bounds $\{\gamma_\mu\}_{\mu\in\mathcal{P}_{numeric}}$ and $\{\gamma_\chi\}_{\mu\in\mathcal{P}_{binary}} > 0$ such that $\theta^*(\lambda)$ is equivalently the solution of the constrained



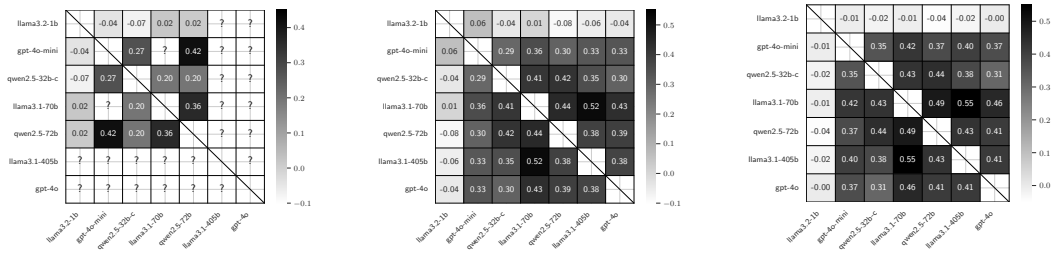
(a) Incorrect Only (b) Correct Only (c) All

Figure .5: MedMCQA: Kendall's τ rank correlations of Llama3, GPT-4o, and Qwen2.5 models ordered by size.



(a) Incorrect Only (b) Correct Only (c) All

Figure .6: TriviaQA: Kendall's τ rank correlations of Llama3 models ordered by size.

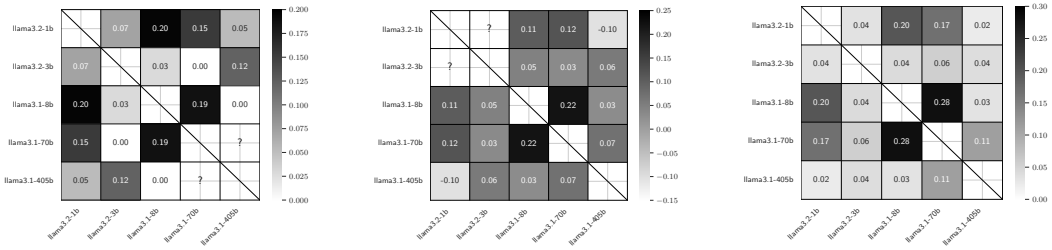


(a) Incorrect Only (b) Correct Only (c) All

Figure .7: TriviaQA: Kendall's τ rank correlations of Llama3, GPT-4o, and Qwen2.5 models ordered by size.

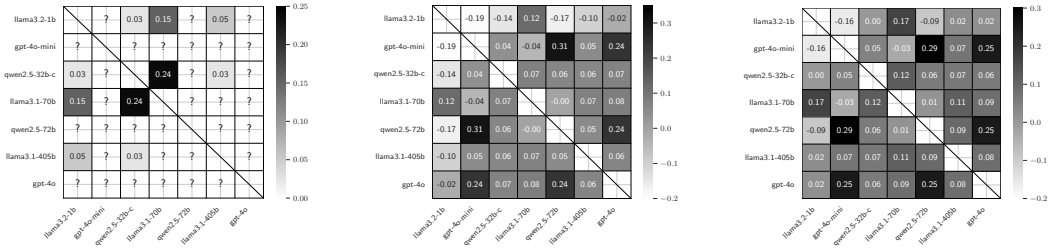
optimization problem

$$\begin{aligned} \theta^* = \mathit{argmin}_\theta \quad & \hat{\mathbb{E}}_\theta[C] \\ \mathit{subject\ to} \quad & \hat{\mathbb{E}}_\theta[\mu] \leq \gamma_\mu, \ \ \mu \in \mathcal{P}_{\mathit{numeric}} \\ & \hat{\mathbb{E}}_\theta[\mathbb{1}_\chi] \leq \gamma_\chi, \ \ \chi \in \mathcal{P}_{\mathit{binary}}, \end{aligned} \tag{12}$$



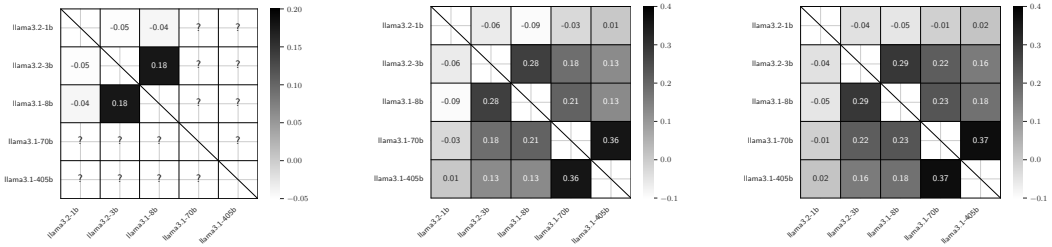
(a) Incorrect Only (b) Correct Only (c) All

Figure .8: XSum: Kendall's τ rank correlations of Llama3 models ordered by size.



(a) Incorrect Only (b) Correct Only (c) All

Figure .9: XSum: Kendall's τ rank correlations of Llama3, GPT-4o, and Qwen2.5 models ordered by size.



(a) Incorrect Only (b) Correct Only (c) All

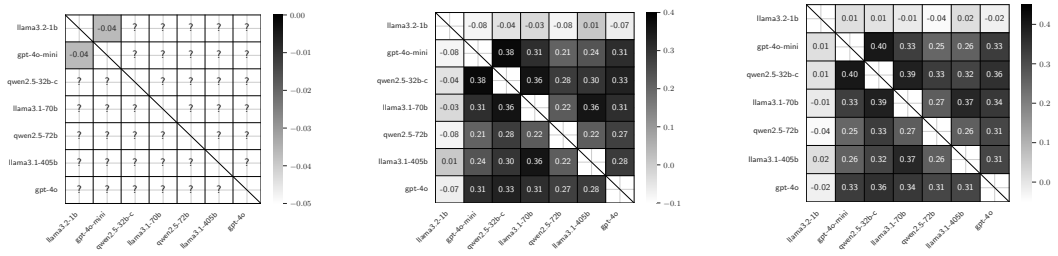
Figure .10: GSM8K: Kendall's τ rank correlations of Llama3 models ordered by size.

and vice versa for $\gamma \mapsto \lambda(\gamma)$*. Then the vector of economic costs,* λ*, maps surjectively onto the Pareto surface via the mapping*

$$\lambda \mapsto (\hat{\mathbb{E}}_{\theta^*(\lambda)}[C], \hat{\mathbb{E}}_{\theta^*(\lambda)}[\mu_1], ..., \hat{\mathbb{E}}_{\theta^*(\lambda)}[\mu_{|\mathcal{P}_{numeric}|}], \hat{\mathbb{P}}_{\theta^*(\lambda)}[\chi_1], ..., \hat{\mathbb{P}}_{\theta^*(\lambda)}[\chi_{|\mathcal{P}_{binary}|}]). \tag{13}$$

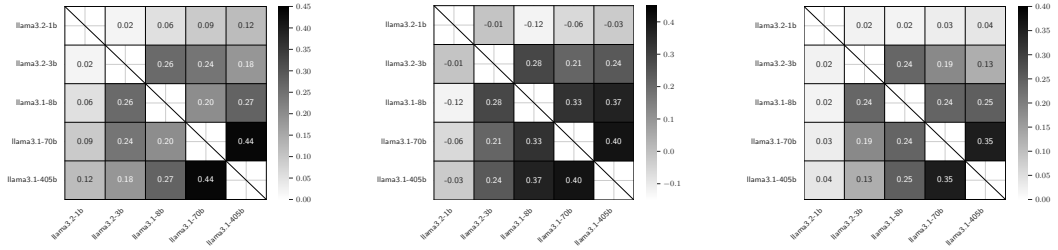
Proof. First, we show that λ maps to the Pareto surface. Second, we show that this mapping is surjective.

Consider any $\lambda \in \mathbb{R}_{>0}^{|\mathcal{P}_{\text{numeric}}|+|\mathcal{P}_{\text{binary}}|}$, and let $x = (\hat{\mathbb{E}}_{\theta^*(\lambda)}[C], ..., \hat{\mathbb{P}}_{\theta^*(\lambda)}[\chi_{|\mathcal{P}_{\text{binary}}|}])$. Suppose for the sake of contradiction that x is not Pareto optimal. Then there exist



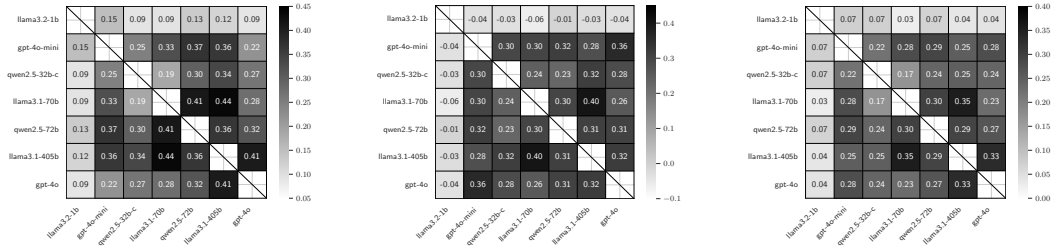
(a) Incorrect Only (b) Correct Only (c) All

Figure .11: GSM8K: Kendall's τ rank correlations of Llama3, GPT-4o, and Qwen2.5 models ordered by size.



(a) Incorrect Only (b) Correct Only (c) All

Figure .12: TruthfulQA: Kendall's τ rank correlations of Llama3 models ordered by size.



(a) Incorrect Only (b) Correct Only (c) All

Figure .13: TruthfulQA: Kendall's τ rank correlations of Llama3, GPT-4o, and Qwen2.5 models ordered by size.

θ' and $x' = (\hat{\mathbb{E}}_{\theta'(\lambda)}[C], ..., \hat{\mathbb{P}}_{\theta'(\lambda)}[\chi_{|\mathcal{P}_{\text{binary}}|}])$ such that x' dominates x. It follows that

$$R(\lambda;\theta') - R(\lambda;\theta^*(\lambda)) = \mathbb{E}_{\theta^*(\lambda)}[C] - \mathbb{E}_{\theta'}[C] \tag{14}$$

$$+ \sum_{\mu} \lambda_\mu (\mathbb{E}_{\theta^*(\lambda)}[\mu] - \mathbb{E}_{\theta'}[\mu]) + \sum_{\chi} \lambda_\chi (\mathbb{P}_{\theta^*(\lambda)}(\chi) - \mathbb{P}_{\theta'}(\chi)) \tag{15}$$

$$> 0, \tag{16}$$

contradicting the optimality of $\theta^*(\lambda)$.

Table .2: Rank correlations of calibrated confidences between Llama3 1B, 3B, 8B, 70B, and 405B models, computed separately for *incorrectly* answered queries ("inc"), *correctly* answered queries ("corr"), and *all* queries ("all"): $\overline{\tau}$ is the average rank correlation across the 10 model pairs; $\tau_{a,b}$ is the rank correlation between data subsets (inc, corr, all) across model pairs; and $p_{a,b}$ is the p value for $\tau_{a,b}$.

Benchmark	$\overline{\tau}_{\text{inc}}$	$\overline{\tau}_{\text{corr}}$	$\overline{\tau}_{\text{all}}$	$\tau_{\text{inc,corr}}$	$p_{\text{inc,corr}}$	$\tau_{\text{inc,all}}$	$p_{\text{inc,all}}$	$\tau_{\text{corr,all}}$	$p_{\text{corr,all}}$
MMLU	0.199	0.396	0.385	0.505	< 0.0001	0.717	< 0.0001	0.686	< 0.0001
MedMCQA	0.174	0.295	0.271	0.324	0.0055	0.486	< 0.0001	0.730	< 0.0001
TriviaQA	0.110	0.274	0.298	0.371	0.0014	0.413	0.0004	0.819	< 0.0001
XSum	0.138	0.049	0.065	0.180	0.1284	0.187	0.1148	0.721	< 0.0001
GSM8K	0.199	0.169	0.201	0.425	0.0003	0.467	< 0.0001	0.921	< 0.0001
TruthfulQA	0.222	0.196	0.178	0.667	< 0.0001	0.860	< 0.0001	0.756	< 0.0001

Now consider any $x \in \mathbb{R}^{|\mathcal{P}_{\text{numeric}}|+|\mathcal{P}_{\text{binary}}|+1}$ on the Pareto surface. Let θ_c^* be the solution of the constrained optimization problem (12) with $\{\gamma_\mu\}$, $\{\gamma_\chi\}$ equal to the last $|\mathcal{P}_{\text{numeric}}| + |\mathcal{P}_{\text{binary}}|$ components of x. Let $x^* = (\hat{\mathbb{E}}_{\theta_c^*}[C], ..., \hat{\mathbb{P}}_{\theta_c^*}[\chi_{|\mathcal{P}_{\text{binary}}|}])$.

We argue that $x = x^*$. Indeed, x is a feasible point of (12) since it lies on the Pareto surface and meets the inequality constraints. In addition, $x_0^* = \hat{\mathbb{E}}_{\theta_c^*}[C]$ cannot be *less* than x_0, since otherwise x^* would dominate x. Furthermore, $x_0^* = \hat{\mathbb{E}}_{\theta_c^*}[C]$ cannot be *greater* than x_0 by the optimality of x^*, since x is feasible. So $x_0^* = x_0$. Hence, $x^* \leq x$ componentwise. Hence, the remaining components of x^* and x must be equal, since otherwise x^* would dominate x. So $x^* = x$.

Now observe that x^* arises from solving (3.18) with $\lambda = \lambda(\gamma)$. Hence, λ maps to x under the mapping (13). □

Appendix 3C — Models and Pricing

Table .3 lists the large language models (LLM) we used in our experiments and the API prices at the time of our experiments (June 2025). We provide each model's exact API identifier together with the cost for input and output tokens.

We used the default hyperparameters (temperature, top-p, top-k, etc.) for sampling from the LLMs, except that we raised the maximum number of output tokens to 100,000. Only for Llama3.3 70B did we implement a lower output token limit of 5,000, since the model otherwise gets caught in endless repetitions on some queries.

We note that we measured negligible roundtrip latency (less than 300ms) to both API endpoints.

Table .3: Details on API providers, LLM identifiers, and costs.

API Provider	Model Identifier	Input $ / M tok	Output $ / M tok
Models accessed through the OpenAI API			
OpenAI	`gpt-4.1-2025-04-14`	2.00	8.00
OpenAI	`o3-2025-04-16`	2.00	8.00
Models accessed through the Fireworks API			
Fireworks	`deepseek-r1-0528`	3.00	8.00
Fireworks	`qwen3-235b-a22b`	0.22	0.88
Fireworks	`llama-v3p3-70b-instruct`	0.90	0.90
Fireworks	`llama-v3p1-405b-instruct`	3.00	3.00

Appendix 3D — Cascading Setup

Similar to Zellinger and Thomson, 2025b, we use self-verification (also known as *P(True)*, from Kadavath et al., 2022) to estimate an LLM's confidence to correctly answer a query. Specifically, given a query, the LLM sends itself a follow-up verification prompt asking whether the proposed answer is correct. Since the response to this query is a single token (Yes/No), we extract the estimated probability of correctness p directly from the LLM's auto-regressive next-token probability. This p is the *self-verification correctness probability*.

To select the optimal confidence threshold for a use case λ, we maximize the cascade's expected reward $R(\lambda;\theta)$ for θ ranging over all 2.5% quantiles of empirically observed self-verification correctness probabilities on the training set. To evaluate the cascade's performance, we fix the optimal confidence thresholds $\theta^* = \theta^*(\lambda)$ (dependent on λ) and compute the expected rewards $R(\lambda;\theta^*(\lambda))$ on the test set.

Appendix 3E — Prompt Templates

This appendix reproduces verbatim the prompt templates used in our experiments. Placeholders are printed in **bold** and are wrapped in curly braces.

MATH Benchmark — Problem-Solving Prompts

System prompt:

```
Your task is to solve a math problem. First think step-by-step,
then end by giving your final answer in the form
'Final Answer: x', where x is the final answer.
DO NOT say anything after that. Make sure to end on the numeric
    answer.
```

User prompt:

```
Your task is to solve the following math problem: {problem}

Reason step-by-step, then give your final by saying
'Final Answer: x', where x is the final numeric answer.
DO NOT say anything after that. Make sure to end on the numeric
    answer.
```

MATH Benchmark — Evaluation Prompts

System prompt:

```
Your task is to determine if an AI model's solution to a
college-level math problem is correct.
If the solution is correct, output "Y".
Otherwise, output "N".
Only output "Y" or "N", nothing else.
```

User prompt:

```
Consider a proposed solution to the following math problem:

Problem:
{problem}

Proposed solution:
{proposed_sol}

Decide if the proposed solution is correct.
Only output "Y" or "N", nothing else.

Correct?
```

Appendix 4A — More Detail on Max-Min Parameters

We give more detail on how `repliclust` manages various geometric attributes using max-min parameters. Table .4 lists all geometric attributes managed with max-min sampling and names the corresponding parameters in `repliclust`. The *reference value* for each geometric parameter serves as a location constraint, while the *max-min ratio* determines the spread. A *constraint* ensures that the distribution of geometric parameters within a data set is similar across data sets drawn from the same archetype.

Table .4: Summary of geometric attributes managed with max-min sampling. The second and third columns indicate whether each max-min ratio or reference value is inferred, or specified by the user as a `parameter`. The fourth column gives the location constraint used during max-min sampling. The *group size* of a cluster is the number of data points in it; the *aspect ratio* is the ratio of the lengths of the longest cluster axis to the shortest. For cluster volumes, we specify the reference value and max-min ratio in terms of *radius* (`dim`-th root of volume) since volumes grow rapidly in high dimensions.

Geometric Attribute	Max-Min Ratio	Reference Value	Constraint
cluster volumes	`radius_maxmin`	`scale`	cluster volumes average to reference volume
group sizes	`imbalance_ratio`	average group size	group sizes sum to number of samples
cluster aspect ratios	`aspect_maxmin`	`aspect_ref`	geometric mean of aspect ratios equals reference
cluster axis lengths	aspect ratio of the cluster	`dim`-th root of cluster volume	geometric mean of lengths equals reference length

To enforce the attribute-specific constraints, `repliclust` samples new values of a geometric parameter in pairs. The first value is randomly drawn from a triangular distribution whose mode and endpoints are determined from the typical value and max-min ratio. The second value is then deterministically computed to maintain the constraint. For reference, see the `sample_with_maxmin` function in the `maxmin.utils` module (Version 1.0.0 of `repliclust`).

Appendix 4B — Attributes of a Mixture Model

Table .5 lists the formal attributes of a mixture model in `repliclust`. A data set archetype provides a way to randomly sample mixture models with similar overall geometric characteristics. Thus, an archetype implicitly defines a probability distribution over the attributes in Table .5.

Table .5: Formal attributes of a mixture model in `repliclust`.

Attribute	Meaning	Mathematical Definition
cluster centers	the positions of cluster centers in space	$\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, ..., \boldsymbol{\mu}_k \in \mathbb{R}^p$
principal axis orientations	the spatial orientation of each cluster's ellipsoidal shape (different for each cluster)	orthonormal matrices $\mathbf{U}_1, \mathbf{U}_2, ..., \mathbf{U}_k \in \mathbb{R}^{p \times p}$
principal axis lengths	the lengths of each cluster's principal axes (axes have different lengths between and within clusters)	$\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2, ..., \boldsymbol{\sigma}_k \in (\mathbb{R}^{>0})^p$
cluster distributions	multivariate probability distributions for generating data (different for each cluster)	distributions $\mathbb{P}_1, \mathbb{P}_2, ..., \mathbb{P}_k$

Appendix 4C — Proof of Theorem 6

We prove Theorem 6 of Section 5.3. Additionally, we provide an analogous result for the simpler "center-to-center" approximation of cluster overlap.

Theorem (LDA-Based Cluster Overlap)**.** *For two multivariate normal clusters with means $\boldsymbol{\mu}_1 \neq \boldsymbol{\mu}_2$ and covariance matrices $\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2$, the approximate cluster overlap α_{LDA} based on the linear separator $\boldsymbol{a}_{LDA} = (\frac{\boldsymbol{\Sigma}_1+\boldsymbol{\Sigma}_2}{2})^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)$ is*

$$\alpha_{LDA} = 2\Big(1 - \Phi\Big(\frac{\boldsymbol{a}_{LDA}^\top(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)}{\sqrt{\boldsymbol{a}_{LDA}^\top \boldsymbol{\Sigma}_1 \boldsymbol{a}_{LDA}} + \sqrt{\boldsymbol{a}_{LDA}^\top \boldsymbol{\Sigma}_2 \boldsymbol{a}_{LDA}}}\Big)\Big), \tag{17}$$

where $\Phi(z)$ is the cumulative distribution function of the standard normal distribution. Moreover, if $\boldsymbol{\Sigma}_1 = \lambda \boldsymbol{\Sigma}_2$ for some λ then α_{LDA} equals the exact cluster overlap α.

Proof. Let $\boldsymbol{a}_{\text{LDA}}$ be the classification axis. Minimax optimality requires that the cluster-specific misclassification probabilities are equal. Since $\boldsymbol{a}_{\text{LDA}}$ is the classification axis, these probabilities correspond to the tails of the marginal distributions along $\boldsymbol{a}_{\text{LDA}}$. Specifically, let

$$\sigma_1 = \sqrt{\boldsymbol{a}_{\text{LDA}}^\top \Sigma_1 \boldsymbol{a}_{\text{LDA}}} \tag{18}$$

be the standard deviation of cluster 1's marginal distribution along $\boldsymbol{a}_{\text{LDA}}$, where Σ_1 is the cluster's covariance matrix; σ_2 is defined analogously. If $\boldsymbol{a}_{\text{LDA}}$ is oriented to point from cluster 1 to cluster 2, then the $1 - \alpha/2$ quantile of cluster 1's marginal distribution meets the $\alpha/2$ quantile of cluster 2's marginal distribution at the decision boundary, where α is the unknown cluster overlap. This intersection implies

$$\boldsymbol{\mu}_1^\top \boldsymbol{a}_{\text{LDA}} + q_{1-\alpha/2}\sigma_1 = \boldsymbol{\mu}_2^\top \boldsymbol{a}_{\text{LDA}} + q_{\alpha/2}\sigma_2, \tag{19}$$

where q_ξ is the ξ-quantile of the standard normal distribution. Rearranging this equation, and using $q_{\alpha/2} = -q_{1-\alpha/2}$ and $\Phi(q_\xi) = \xi$, gives (17).

Next, suppose that $\boldsymbol{\Sigma}_1 = \lambda \boldsymbol{\Sigma}_2$ for some λ. In this case, maximum likelihood classification results in a linear decision boundary that coincides with the LDA solution. Hence, the minimax-optimal linear classifier uses the LDA-based classification axis $\boldsymbol{a}_{\text{LDA}}$. □

Theorem (Center-to-Center Cluster Overlap)**.** *For two multivariate normal clusters with means $\boldsymbol{\mu}_1 \neq \boldsymbol{\mu}_2$ and covariance matrices $\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2$, the center-to-center cluster*

overlap α_{C2C}, based on a classification boundary perpendicular to the line connecting the cluster centers, is

$$\alpha_{C2C} = 2\Big(1 - \Phi\Big(\frac{\boldsymbol{\delta}^\top \boldsymbol{\delta}}{\sqrt{\boldsymbol{\delta}^\top \boldsymbol{\Sigma}_1 \boldsymbol{\delta}} + \sqrt{\boldsymbol{\delta}^\top \boldsymbol{\Sigma}_2 \boldsymbol{\delta}}}\Big)\Big), \tag{20}$$

where $\boldsymbol{\delta} := \boldsymbol{\mu}_2 - \boldsymbol{\mu}_1$ is the difference between cluster centers and $\Phi(z)$ is the cumulative distribution function of the standard normal distribution.

Moreover, if the covariance matrices $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$ are both multiples of the identity matrix, then α_{C2C} equals the exact cluster overlap α.

Proof. The proof proceeds along the same lines as the proof of Theorem 6, except that the classification axis is $\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1$. If both covariance matrices are multiples of the identity matrix, $\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1$ is a scalar multiple of the LDA-based classification axis $\boldsymbol{a}_{\text{LDA}}$. Hence, the second part of Theorem 6 kicks in to establish equality between α_{C2C} and the exact overlap. □

Appendix 4D — Overlap Control for Non-Gaussian Clusters

Figure .15 visualizes two-dimensional data sets created from the same archetype but with different radial probability distributions. The results suggest that pegging the 68.15% quantile of the radial distribution at unity leads to satisfactory overlap control for distributions with infinite support. However, for distributions with bounded support (such as the beta distribution), this approach leads to greater separation between the clusters, as shown in the rightmost column of Figure .15.

Appendix 4E — Neural Network Architecture for Distortion

The default architecture for the neural network used in the `distort` function of Section 5.3 is a feed-forward network consisting (in order) of a linear embedding, 16 repeated feed-forward blocks (each consisting of a fully connected linear layer followed by layer normalization and Tanh activation), and a linear projection. Importantly, we tie the weights of the embedding and projection layers, so that the projection weights are the transpose of the embedding weights (Inan, Khosravi, and Socher, 2017). The default hidden dimensionality is 128, so that the embedding layer maps a data set archetype's dimension to 128. The internal feed-forward blocks preserve this hidden dimension, and the final projection layer maps it back to the archetype's dimension.

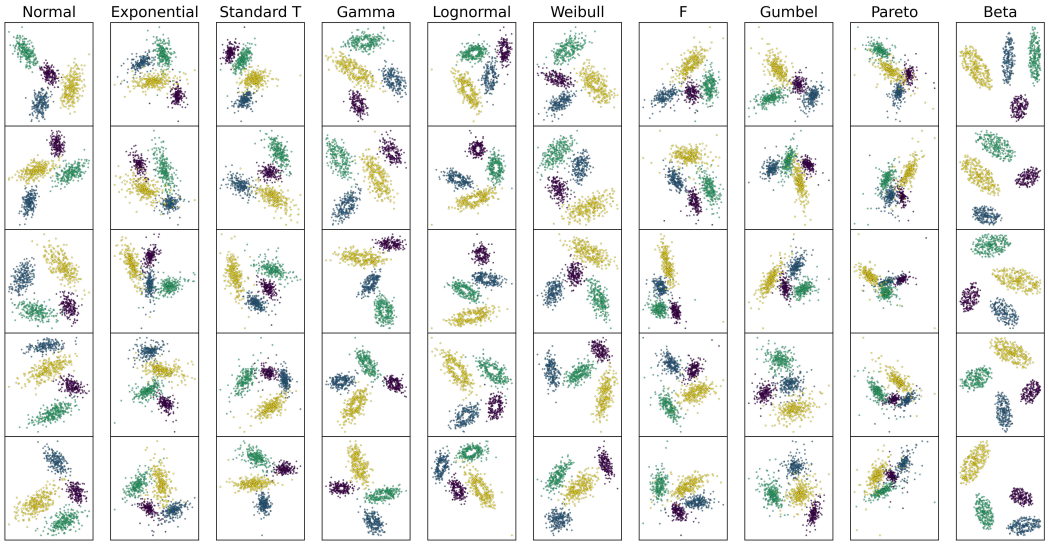


Figure .15: Overlap control works well for non-normal probability distributions with infinite support. The data sets shown are generated from the same archetype in 2D (with overlap at around 1%), except that we change the probability distribution in each column. The only distribution that violates 1% overlap is the beta distribution, which unfortunately generalizes to other distributions with bounded support. Note that the heavy-tailed distributions (Pareto, F, ...) appear smaller on scatter plots because they give rise to outliers.

Appendix 4F — Prompt Templates

Below we list the prompt templates (including few-shot examples) used in Version 1.0.0 of `repliclust`. Up-to-date versions are available in the code base (see `repliclust.org`).

1. Prompt template mapping archetype description to high-level geometric parameters:

```
Your task is to turn a verbal description of a data set archetype from
    Repliclust into a precise JSON that specifies which parameter settings
    to use to create the desired data set archetype in Repliclust. These
    are the allowed parameters:


n_clusters: int >= 1, the number of clusters to generate
dim: int >= 2, the dimensionality of the data
n_samples: int >= 1, the number of data samples to generate
aspect_ref: float >= 1, the eccentricity of a typical cluster (how oblong
    vs spherical it is)
aspect_maxmin: float >= 1, how much the eccentricity varies across
    clusters in a data set
```

```
radius_maxmin: float >= 1, how much cluster radius (and thereby cluster
    volume) varies across the clusters
max_overlap: float > 0, the maximum allowed overlap between any pair of
    clusters (0.1-0.2 is significant overlap, 0.01-0.05 is little overlap,
    0.001 is very little overlap, and 0.0001 and lower is well-separated)
min_overlap: float > 0, the minimum amount of overlap each cluster should
    have with some other cluster, preventing a cluster from being too far
    away from all other clusters
imbalance_ratio: float >= 1, specifies how imbalanced the number of data
    points per cluster is
distributions: list[str], determines the probability distributions to use
    for the clusters; the available distributions are 'normal',
    'standard_t', 'exponential', 'beta', 'uniform', 'chisquare', 'gumbel',
    'weibull', 'gamma', 'pareto', 'f', and 'lognormal'

IMPORTANT NOTES:
Any words like "separated", "far away", "close together", or "overlapping"
    refer to the overlap between clusters. Far apart means that
    max_overlap is 1e-4 or less
Always make min_overlap smaller than max_overlap, usually ten times
    smaller!
ONLY include the Pareto ('pareto') distribution if the user specifically
    asks for heavy tails!

EXAMPLES:

Description: five oblong clusters in two dimensions
Archetype JSON: {
  "n_clusters": 5,
  "dim": 2,
  "n_samples": 500,
  "aspect_ref": 3,
  "aspect_maxmin": 1.5,
}

Description: three spherical clusters with significant overlap in two
    dimensions
Archetype JSON: {
  "n_clusters": 3,
  "dim": 2,
  "n_samples": 300,
  "max_overlap": 0.2,
  "min_overlap": 0.1,
```

```
  "aspect_ref": 1.0,
  "aspect_maxmin": 1.0
}
```


Description: eight spherical clusters of different sizes with significant overlap in two dimensions


```
Archetype JSON: {
  "n_clusters": 8,
  "dim": 2,
  "n_samples": 800,
  "max_overlap": 0.25,
  "min_overlap": 0.1,
  "aspect_ref": 1.0,
  "aspect_maxmin": 1.0,
  "radius_maxmin": 2.0
}
```


Description: ten clusters which are all highly oblong (and equally so) but of very different sizes, with moderate overlap


```
Archetype JSON: {
  "n_clusters": 10,
  "n_samples": 1000,
  "aspect_ref": 5,
  "aspect_maxmin": 1.0,
  "max_overlap": 0.10,
  "min_overlap": 0.05,
  "radius_maxmin": 4.0
}
```


Description: five clusters with significant class imbalance


```
Archetype JSON: {
  "n_clusters": 5,
  "n_samples": 500,
  "imbalance_ratio": 5,
  "aspect_ref": 1.5,
  "aspect_maxmin": 1.5
}
```


Description: five clusters with perfect class balance


```
Archetype JSON: {
  "n_clusters": 5,
  "n_samples": 500,
  "imbalance_ratio": 1.0,
```

```
  "aspect_ref": 1.4,
  "aspect_maxmin": 1.6
}

Description: eight clusters of which 70% are exponentially distributed and
    30% are normally distributed
Archetype JSON: {
  "n_clusters": 8,
  "n_samples": 800,
  "aspect_ref": 1.7,
  "aspect_maxmin": 1.5,
  "distributions": ["exponential", "normal"],
  "distribution_proportions": [0.7, 0.3],
}

Description: eight clusters with 1000 total samples of which half are
    exponentially distributed and half are normally distributed
Archetype JSON: {
  "n_clusters": 8,
  "n_samples": 1000,
  "aspect_ref": 1.7,
  "aspect_maxmin": 1.5,
  "distributions": ["exponential", "normal"],
  "distribution_proportions": [0.5, 0.5]
}

Description: two clusters of different sizes in 10 dimensions that are
    well-separated
Archetype JSON: {
  "n_clusters": 2,
  "dim": 10,
  "n_samples": 200,
  "aspect_ref": 2
  "aspect_maxmin": 2,
  "radius_maxmin": 4.0,
  "max_overlap": 0.001,
  "min_overlap": 0.0001
}

Description: very oblong clusters that overlap heavily
Archetype JSON: {
  "n_clusters": 6,
  "n_samples": 600,
```

```
  "aspect_ref": 7,
  "aspect_maxmin": 1.4,
  "max_overlap": 0.4,
  "min_overlap": 0.3
}

Description: highly separated and very oblong clusters
Archetype JSON: {
  "n_clusters": 4,
  "n_samples": 400,
  "aspect_ref": 6,
  "aspect_maxmin": 1.6,
  "max_overlap": 1e-4,
  "min_overlap": 1e-5
}

Description: ten clusters with very different shapes
Archetype JSON: {
  "n_clusters": 10,
  "n_samples": 1000,
  "aspect_ref": 1.5,
  "aspect_maxmin": 3.0,
  "radius_maxmin": 3.0
}

Description: twelve well-separated clusters with very different shapes
Archetype JSON: {
  "n_clusters": 12,
  "n_samples": 1200,
  "aspect_ref": 1.5,
  "aspect_maxmin": 5.0,
 "radius_maxmin": 5.0,
 "max_overlap": 1e-4,
  "min_overlap": 1e-5
}}

Description: twelve highly separated Gaussian clusters with very different
    shapes
Archetype JSON: {
  "n_clusters": 12,
  "n_samples": 1200,
  "aspect_ref": 1.5,
  "aspect_maxmin": 5.0,
```

```
 "radius_maxmin": 5.0,
 "max_overlap": 1e-4,
  "min_overlap": 1e-5
 "distributions": ["normal"]}}

Description: five heavy-tailed clusters
Archetype JSON: {
  "n_clusters": 5,
  "n_samples": 500,
  "aspect_ref": 1.5,
 "distributions": ["standard_t", "lognormal", "pareto"]}}

Description: clusters with holes
Archetype JSON: {"distributions": ["f"]}

Description: clusters from a variety of distributions
Archetype JSON: {"distributions": ["normal", "exponential", "gamma",
    "weibull", "lognormal"]}

Description: clusters from all different distributions
Archetype JSON: {"distributions": ['normal', 'standard_t', 'exponential',
    'beta', 'uniform', 'chisquare', 'gumbel', 'weibull', 'gamma', 'f', and
    'lognormal']}

Description: clusters from different distributions
Archetype JSON: {"distributions": ['normal', 'exponential', 'beta',
    'uniform', 'chisquare', 'gumbel', 'weibull', 'gamma', 'f', and
    'lognormal']}

Description: highly separated clusters from all different distributions
    but no heavy tails
Archetype JSON: {"max_overlap": 1e-4,
  "min_overlap": 1e-5,
 "distributions": ['normal', 'exponential', 'beta', 'uniform',
    'chisquare', 'gumbel', 'weibull', 'gamma', 'f', and 'lognormal']}

Description: seven clusters with uniform distribution with light overlap
Archetype JSON: { "max_overlap": 0.025,
"min_overlap": 0.0025,
 "distributions": ["uniform"]}

Description: clusters with bounded support
Archetype JSON: {"distributions": ["beta", "uniform"]}
```

```
Description: {description}
Archetype JSON:
```

2. Prompt template mapping archetype description to a descriptive identifier:

```
Your task is to turn a description of a data set archetype into an
    identifier for
the archetype. The identifier should be short yet descriptive, and not
    contain any whitespace
(but underscores are OK). IMPORTANT: the identifier should be a valid
    Python variable name.
Specifically, it may NOT start with a number, nor contain any special
    character except for
underscores.

EXAMPLES:

Description: five oblong clusters in two dimensions
Archetype identifier: five_oblong_2d

Description: three spherical clusters with significant overlap in two
    dimensions
Archetype identifier: three_spherical_significant_overlap_2d

Description: eight spherical clusters of different sizes with significant
    overlap in two dimensions
Archetype identifier:
    eight_spherical_different_sizes_significant_overlap_2d

Description: ten clusters which are all highly oblong (and equally so) but
    of very different sizes, with moderate overlap
Archetype identifier:
    ten_highly_oblong_very_different_shapes_moderate_overlap

Description: five clusters with significant class imbalance
Archetype identifier: five_significant_class_imbalance

Description: five clusters with perfect class balance
Archetype identifier: five_perfect_class_balance

Description: eight clusters of which 70% are exponentially distributed and
    30% are normally distributed
```

```
Archetype identifier: eight_exp_and_norm

Description: eight clusters with 1000 total samples of which half are
    exponentially distributed and half are normally distributed
Archetype identifier: eight_exp_and_norm_1000_samples

Description: two clusters of different sizes in 10 dimensions that are
    well-separated
Archetype identifier: two_different_sizes_well_separated_10d

Description: very oblong clusters that overlap heavily
Archetype identifier: very_oblong_heavy_overlap

Description: ten clusters with very different shapes
Archetype identifier: ten_very_different_shapes

Description: {description}
Archetype identifier:
```