AutoMix: Automatically Mixing Language Models

Anonymous ACL submission

Abstract

Large language models (LLMs) are now available from cloud API providers in various sizes and configurations. While this diversity offers a broad spectrum of choices, effectively leveraging the options to optimize computational cost and performance remains challenging. In this work, we present `AutoMix`, an approach that strategically routes queries to larger LMs, based on the approximate correctness of outputs from a smaller LM. Central to AutoMix is a few-shot self-verification mechanism, which estimates the reliability of its own outputs without requiring training. Given that verifications can be noisy, we employ a meta-verifier in AutoMix to refine the accuracy of these assessments. Our experiments using LLAMA2-13/70B, on five context-grounded reasoning datasets demonstrate that AutoMix surpasses established baselines, improving the incremental benefit per cost by up to 89%.[1]

1 Introduction

Human problem-solving inherently follows a multi-step process: generate a solution, verify its validity, and refine it further based on verification outcomes. The emulation of this self-refinement and reflective behavior has gained attention in the recent research (Pan et al., 2023a; Madaan et al., 2023; Reid and Neubig, 2022; Schick et al., 2022; Welleck et al., 2022; Shinn et al., 2023). Classic self-refine paradigms consistently employ a single model across all problem-solving stages, demonstrating effectiveness in specific scenarios (Madaan et al., 2023; Shinn et al., 2023). Yet, the intrinsic complexity and variability of tasks, from simplistic (e.g., binary classification on separable data) to complex (e.g., code generation) and potentially unsolvable (e.g., certain forms of multi-step reasoning), motivate an alternative approach of *model switching*. Model switching iteratively queries over models of disparate sizes and capabilities, verifying feedback at each step and determining whether to accept the output or route to a more capable, albeit computationally intensive, model (Liu et al., 2020; Zhou et al., 2020; Madaan and Yang, 2022; Geng et al., 2021; Schuster et al., 2022).

Past studies in model-switching strategies predominantly rely on separate models trained explicitly for each step or require access to logits(Chen et al., 2023; Welleck et al., 2022; Reid and Neubig, 2022). However, modern LLM often provide access solely through black-box APIs, restricting direct model optimization and adaptability due to the unavailability of fine-tuning capabilities and weight access. In response, we introduce `AutoMix`, a method that utilizes black-box LLM APIs, circumventing the necessity for separate models or logits access by adopting few-shot learning strategies (Brown et al., 2020) and implementing self-verification. Our method proposes strategies for each step of problem-solving: solution generation, verification, and routing, all assuming we only have access to black-box LLMs.

In contrast to existing approaches, which generally classify tasks as Simple or Complex for model routing (Chen et al., 2023), `AutoMix` integrates a third category of *Unsolvable* queries. These queries are too complex to be solved even by a Large Language Model (e.g., due to underspecification) and should not be routed to larger models if identified early. This consideration allows `AutoMix` to judiciously allocate computational resources, avoiding cases where resources are wasted on these particularly challenging instances.

We use context-grounded few-shot entailment to evaluate the consistency of generated answers with the provided context, without requiring a large amount of human-labeled data (Poliak, 2020; Dagan et al., 2022). For example, an answer discussing "desert animals" in a context focused on "aquatic life" would be flagged as inconsistent.

[1] We will release the code and data upon acceptance.

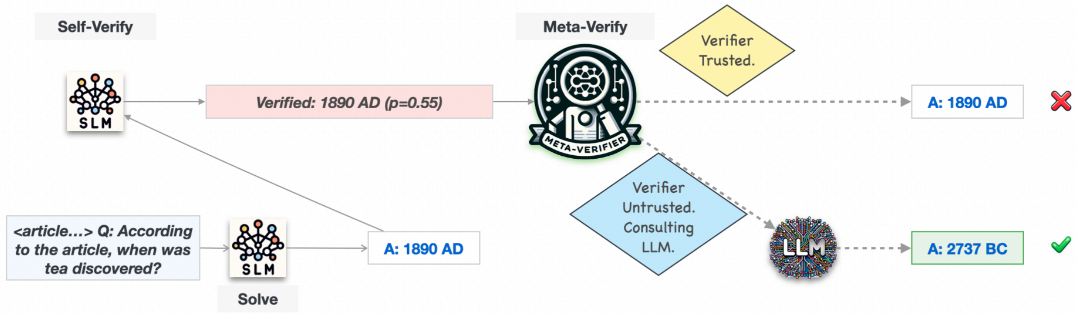

Figure 1: AutoMix: Given a context (like an article) and a question q, an initial answer (*1890 AD*) is generated with the smaller language model (SLM). The answer is self-verified by the SLM, yielding a noisy verification score. The Meta-Verifier subsequently assesses verifier's results. Based on the meta-verifier's decision, either the initial answer (*1890 AD*) is returned, or the question is rerouted to a larger language model (LLM) to enhance accuracy.

However, recognizing that self-verification can sometimes be inconsistent or noisy (Huang et al., 2023), we introduce a *meta-verifier* to evaluate the reliability of the initial verification. The meta-verifier acts as a secondary check, providing an additional layer of confidence assessment to ensure that the decision to route a task to a larger or smaller model is well-founded.

In summary, our contributions are:

- We introduce AutoMix, a method that strategically leverages black-box LLM APIs for generating a solution, verifying the solution, and switching to a larger language model, everything without access to model weights, gradients, or logits.
- We also show that context-grounded entailment is a reasonable but noisy proxy for self-verification. To deal with this noise, we propose a POMDP-based meta-verification mechanism that helps improve the reliability of the final decision.
- We propose and introduce the *Incremental Benefit Per Unit Cost* (IBC) metric, a novel measure that quantifies the efficiency of integrating smaller and larger language models.
- We present empirical evidence from experiments on five context-grounded reasoning datasets using the language models LLAMA2-13B and LLAMA2-70B as the small (SLM) and large (LLM) language models. Our results demonstrate that AutoMix surpasses baselines, enhancing the incremental benefit per cost by up to 89%.

2 AutoMix: Few-shot Self-Verification and Meta-Verification

```
Context: {context}

Question: {question}

AI Generated Answer: {generated_answer}

Instruction: Your task is to evaluate
↪ if the AI Generated Answer is
↪ correct, based on the provided
↪ context and question. Provide the
↪ judgement and reasoning for each
↪ case. Choose between Correct or
↪ Incorrect.

Evaluation:"
```

Listing 1: **Verification Prompt.** The verification process is framed as a natural language entailment task, where the model determines the validity of the model-generated answer with respect to the context and question. We use a generic few-shot prompt for all tasks (prompt in appendix E.1).

Task and setup We tackle the problem of context-grounded question answering, where given a context $\mathcal{C}$ (e.g., stories, newswire, or research article) and a question q, the model is tasked with generating an accurate and coherent answer, consistent with the provided context. Our choice of tasks is motivated by two key concerns: (1) longer queries are more computationally demanding, underscoring the need for an approach like AutoMix to navigate the cost-accuracy trade-off, and (2) the context allows for cross-checking preliminary answers with available information using self-verification (described shortly). While self-verification in reasoning tasks is challenging for LLMs (Pan et al., 2023a; Huang et al., 2023), we find that context significantly aids this process.

We deploy two distinct models: a smaller, cost-efficient model, denoted as SLM (smaller language model), and a larger, more accurate yet costly model, LLM (large language model. Our objective is to optimize performance while staying economical. An initial answer, $\mathcal{A}_s$, is generated using the smaller SLM. Further, in Appendix B.2 we extend `AutoMix` to 3 models by incorporating medium language model, showing significant gains.

Few-shot Verification To assess the trustworthiness of $\mathcal{A}_s$, we employ a few-shot verifier, $\mathcal{V}$, which ascertains the validity of SLM's outputs and decides if a query should be redirected to LLM. Different from existing works that perform verification by creating a new question (Weng et al., 2022; Jiang et al., 2023), we frame verification as an entailment task (Dagan et al., 2005; Poliak, 2020; Dagan et al., 2022), aiming to determine if the answer generated by SLM aligns with the provided context. Specifically, the verifier gauges $v = \text{p}(\text{correct} = 1 \mid \mathcal{A}_s, \mathcal{C}, q)$, with correct $= 1$ indicating that $\mathcal{A}_s$ is correct. The verification prompt is outlined in Figure 1. We use the same verification prompt for all tasks. Figure 2 shows an example.

2.1 Meta-verifier

Given the potential inconsistency or noise in verifier outcomes, a secondary evaluation mechanism, which we term the *meta-verifier*, is crucial to vet the verifier's conclusions. In particular, the verifier is tasked with determining whether the SLM's answer is entailed by the context, and this decision is made without considering the inherent difficulty of the problem. Notably, routing *Unsolvable* queries to the LLM is resource-inefficient and does not enhance performance. While ascertaining the ground truth of query difficulty is non-trivial, verification probability and trends from historical data inferred using validation set, can provide insightful guidance. Formally, we define the meta-verifier's outputs as $m(v, \mathcal{A}_s, \mathcal{C}, q) \rightarrow \{0, 1\}$, where $m = 1$ implies the verifier's output can be trusted.

Addressing the notable challenges of self-correction in large language models (Madaan et al., 2023; Huang et al., 2023), our method employs a non-LLM setup for meta-verification to avoid escalating issues like hallucination and reasoning errors (Dziri et al., 2023). The meta-verifier can adopt various learning strategies, including supervised learning, reinforcement learning, and symbolic reasoning, explored further in upcoming sections. Subsequent sections provide a deeper exploration into two particular implementations of this strategy.

Thresholding In this simplistic meta-verifier approach, the decision is made based on probability of verifier being correct with a threshold t, defined as $m_t(v) = 0$ for $v < t$ and $m_t(v) = 1$ for $v \geq t$. Intuitively, a high probability of verifier indicates that verifier is confident in it's decision and can be trusted. For black-box language models, the probability of correctness can be derived by sampling $k > 1$ samples at a higher sampling temperature.

Context

Which country grows the most tea? The answer is India. It grows three times as much as China. Which country drinks the most tea? It's neither China nor Japan. It's Great Britain. [...] In general, there are two kinds of tea. Black tea and green tea. [...] One customer put the bag into a pot. Then he just poured hot water over it. And the tea bag was born. **Shen Nong was the first to drink tea. (Shen was a Chinese emperor.) This was about 2737 B.C.** Shen had bad digestion. So he drank several cups of hot water daily[...] Tea became the drink of China.

Question

When did Shen Nong drink tea?

Generated Answer (by LLAMA2-13B)

He took it in 1990.

Verifier Output (by LLAMA2-13B)

The context does not mention that he took tea in 1990. The AI generated answer is **Incorrect.**

Figure 2: **Context-Grounded Self-Verification in Action.** The example showcases the verifier, utilizing the *same model* as the answer generator, identifying and rejecting an inaccurate answer—*He took it in 1990*—by effectively leveraging the context. The example uses LLAMA2-13B for both generation and verification on a COQA dataset instance.

Using a POMDP In the context of a meta-verifier, we observe that the queries could be categorized into three different categories: *Simple*, *Complex*, and *Unsolvable*. The simple queries are addressable by SLM itself; the complex queries are addressable by LLM but not by SLM, and Unsolvable queries are so complex that they cannot be solved by LLM or SLM. Hence, a ground truth oracle should route only the complex queries but not unsolvable queries. Since the ground truth state, i.e., the query category is unknown and unobserved, we formulate this decision problem as a Partially Observable Markov Decision Process (POMDP) (Monahan, 1982). POMDP presents a robust framework, offering a structured way to manage and navigate through the decision spaces where the system's state is not fully observable. A POMDP is defined by a tuple (S, A, T, R, Ω, O), where S is a set of states, A is a set of actions, T represents the state transition probabilities, R is the reward function, Ω is a set of observations, and O is the observation function.

In our scenario, the states S correspond to the three question categories: *Simple*, *Complex*, and *Unsolvable*. Actions are denoted as either reporting the SLM answer or routing to the LLM. Observations, in the form of verifier output v, enable the POMDP to ascertain its belief state, which is a probability distribution over S. For instance, a high verifier confidence in the correctness of $\mathcal{A}_s$ would increase the belief in the *Simple* state. The solution to the POMDP subsequently yields a policy that maps belief states to actions, effectively deciding whether to invoke the LLM based on a balance of expected future rewards and computational costs. See Appendix B.1 for more details.

Another advantage of the POMDP-based meta-verifier is its interpretability and customizability via reward assignment. For instance, in a *Complex* state, assigning a very high reward of +50 for invoking the LLM indicates a preference for accurate solutions over computational cost. Although the POMDP framework inherently handles sequences of decisions, we confine our approach to a single-decision scenario (horizon or episode length 1) for simplicity, with the potential for extension to streaming settings for optimizing across multiple queries or a fixed time duration.

3 Cost-Performance Efficiency Analysis

In our approach to leveraging model performance, it is essential to consider not only the raw accuracy of predictions but also the associated computational or monetary costs. To that end, we introduce a metric to understand the efficiency of the models in terms of cost. We use C_M and P_M to denote the cost and performance of a method M. We also use C_{SLM} and C_{LLM}, and P_{SLM} and P_{LLM}, to denote the cost and performance of using the SLM and LLM, respectively.

Incremental Benefit Per Cost (IBC) We introduce methods, denoted by M, to optimally integrate SLM and LLM. For each method M, we associate a cost C_M and performance P_M. To quantify the utility of M over SLM, we define the metric *Incremental Benefit Per Cost* (IBC) as IBC_M (Equation (3)).

$$\text{IBC}_M = \frac{P_M - P_{\text{SLM}}}{C_M - C_{\text{SLM}}}, \tag{1}$$

$$\text{IBC}_{\text{BASE}} = \frac{P_{\text{LLM}} - P_{\text{SLM}}}{C_{\text{LLM}} - C_{\text{SLM}}}, \tag{2}$$

$$\Delta_{\text{IBC}}(M) = \frac{\text{IBC}_M - \text{IBC}_{\text{BASE}}}{\text{IBC}_{\text{BASE}}} \times 100 \tag{3}$$

The IBC metric captures the efficiency of performance enhancement relative to the additional cost. For comparative evaluation, we set a baseline IBC, IBC_{BASE}, representing the benefit of *always* using LLM over SLM. Finally, we compare methods using Δ_{IBC}, which compares the IBC of a specific method with IBC_{BASE}. A positive IBC lift suggests that M achieves performance increments more cost-effectively than a standalone LLM, whereas a negative lift indicates reduced efficiency (Figure 3)

Geometric Interpretation On a Performance vs. Cost plot, consider the line segment joining the data points of small language model (SLM) and large language model (LLM). This segment's slope represents a basic rate of performance increase for each additional unit of cost. The Incremental Benefit per Cost (IBC) for any method M is the slope of the line from the SLM point to the point representing M(Figure 3). A method M that lies above the SLM-LLM segment provides a steeper slope, indicating a favorable IBC (and a positive Δ_{IBC}). Conversely, if M lies below the segment, it suggests an unfavorable or negative IBC. Our primary objective is to identify or develop methods that yield a consistently positive IBC, maximizing performance enhancements for each additional unit of cost.

Cost Calculation To evaluate the efficiency of a method M that leverages both the Small Language Model (SLM) and the Large Language Model (LLM), we define a cost metric, C_M. This metric incorporates the costs of both initial answer generation and verification by the SLM, as well as potential routing to the LLM. Specifically, the total cost C_M is computed as $C_M = 2 \times C_{\text{SLM}} + w_{\text{LLM}} \times C_{\text{LLM}}$. Here, C_{SLM} and C_{LLM} represent the costs of a single query to the SLM and LLM, respectively.

```
procedure ANSWERQUERY(C, q)
        ▷ C: Context, q: Question, SLM/LLM: Small/large language model
    A_s ← SOLVE(SLM, C, q)
    v ← SELF-VERIFY(A_s, C, q)
    if META-VERIFY(v, A_s, C, q) then
        return A_s
    else
        A_l ← SOLVE(LLM, C, q)
        return A_l
    end if
end procedure
```

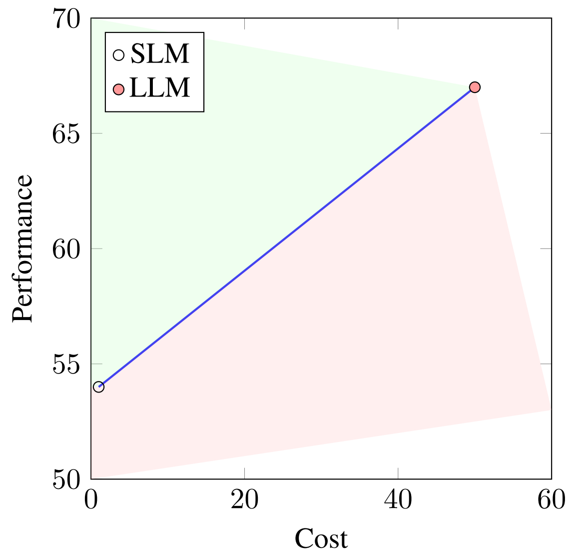

Figure 3: **Left:** `AutoMix` algorithm. **Right**: Performance vs. Cost curve. The slope between SLM and LLM provides a way to the Incremental Benefit per Cost (IBC) for methods that mix models. Methods with a steeper slope than this reference when plotted against SLM have a positive IBC (green region), whereas those below the reference have a negative IBC (red region), falling into the red region.

The factor $w_{\text{LLM}} \in [0, 1]$ denotes the proportion of times the LLM is used, with $w_{\text{LLM}} = 1$ indicating exclusive use and $w_{\text{LLM}} = 0$ denoting no usage. It's important to note that while our framework uses the SLM for verification, alternative verifiers could be incorporated, which would adjust the cost formula accordingly.

While various complexities determine the pricing of these APIs (Dehghani et al., 2021), given our emphasis on black-box utilization of large language models, we choose to represent cost simply: the monetary expense charged to the end user by the language model APIs.

4 Experiments

Setup We experiment with open-source pair LLAMA2-13B and LLAMA2-70B (Touvron et al., 2023). We assume a cost of 1 unit for the SLM, and 50 units for the LLM, following the price disparity between the small and large models offered by LLM API providers like OpenAI and Together [2]. Furthermore, in practical setups, SLM might be deployed with on-premise hardware, and LLM might be only available through relatively expensive APIs, further skewing the cost ratio. Please see Appendix D for more details on the experimental setup.

[2] https://openai.com/pricing, https://together.ai/

Datasets We experiment with several datasets, each with its unique context and evaluation metric: i) NARRATIVE-QA (Kočiský et al., 2018), which involves question answering about full-length books and movie scripts (F1 score); ii) QASPER (Dasigi et al., 2021), focusing on question answering over research papers (F1 score); iii) CNLI (Koreeda and Manning, 2021), which targets natural language inference tasks using non-disclosure agreements as context and evaluates using accuracy; iv) QUALITY (Pang et al., 2022), comprised of multiple-choice questions from long articles and stories, evaluated on exact match; and v) COQA (Reddy et al., 2019), consisting of conversational comprehension questions that test models on coreference and pragmatic reasoning (F1 score). For all datasets, we retain a subset of the context (3500 tokens max) by performing retrieval using the question as the key. We use `all-MiniLM-L6-v2` by Reimers and Gurevych (2019) for retrieval.

For evaluation, we utilize the validation sets from Shaham et al. (2022) for NARRATIVE-QA, QASPER, CNLI, and QUALITY, and use the prompts from Shaham et al. (2023). For COQA, we employ its validation split and adapt the QUALITY prompt. Regardless of dataset, identical input prompts are dispatched to both SLM and potentially LLM, ensuring consistent input processing costs. The output length is fixed in multichoice datasets like CNLI and QUALITY, and the brevity of responses in other datasets allows us to assume uniform output processing costs. We use greedy decoding (temperature 0) and draw a single sample for both the SLM and LLM.

Baselines We use Frugal GPT (F) (Chen et al., 2023) as the baseline. We finetune a DistillBert (Sanh et al., 2019) as a verifier, outputting a confidence probability for a given question, context, and SLM-generated answer, with a verifier confidence threshold directing query routing and its cost set to 0 due to significantly lower operational costs than SLM. Both approaches operate in a low-resource setting, utilizing 1000 training examples per dataset.

Proposed approaches We experiment with three different types of meta-verifiers: i.) **`AutoMix` + Self-Consistency**: This method choses the majority decision from verifier from 8 drawn samples and performs the decision without any explicit meta-verifier. ii) **AutoMix + Thresholding**: Using a threshold on the verifier probability e.g., *Thresh=0.75* implies using SLM outputs with confidence ≥ 0.75 and LLM. We use a threshold for each dataset that yields the highest Δ_{IBC} on the validation set. iii) **AutoMix + POMDP**: This method optimizes routing decisions using a POMDP solver (Smith and Simmons, 2006) as a meta-verifier. The POMDP is learned on the validation set, and takes decision based on the verifier outputs (detailed in Appendix B.1).

4.1 Main Results

Table 1 shows the meta-verifier method consistently showcases superior performance in terms of Δ_{IBC} across both LLAMA2-13/70B. On all datasets, `AutoMix` beat FrugalGPT despite the latter having access to domain-specific training and low verifier cost. Further, on 3 of the 5 datasets, `AutoMix`-POMDP is the best performing method, with positive improvement on all but QASPER. We see maximum gains in COQA and CNLI, with `AutoMix` showing maximum improvement of 56% and 89% respectively. In Figure 4 (left), we present the performance of our model, `AutoMix`, across various cost intervals. Our findings reveal that `AutoMix`-POMDP shows consistent positive Δ_{IBC} across all evaluated costs. This suggests that our method can deliver consistent improvements, regardless of the user's desired cost or performance requirements. Further, in Figure 4 (right), we compare the accuracy of using POMDP based meta-verifier over Verifier-SC. We see significant improvements across all datasets, with relative gains of up to 42% demonstrating our proposed meta-verifier's importance in few-shot verification setups. It's noteworthy that even modest savings in computational cost can translate to significant financial implications at the scale of LLM operations, underscoring the economic relevance of our approach.

5 Analysis

5.1 When and Why does meta-verification help?

Figure 5 shows the relationship between the F1 score improvement b/w LLM and SLM, denoted as $\Delta P_{LLM-SLM}$ (y-axis), for different verifier confidence values (x-axis). Ideally, points with a high $\Delta P_{LLM-SLM}$ should be directed to LLM, as they result in significant F1 score gains. A well-calibrated verifier should exhibit a decreasing linear trend: assign higher confidence to points where the gains from using a LLM are lower. However, this behavior is only observed in the NARRATIVE-QA and COQA datasets. In such scenarios, the need for a robust meta-verifier is reduced as raw outputs from the verifier can be trusted. As a result, self-verification performs well with simple techniques like self-consistency and thresholding.

The verifier exhibits a peculiar behavior on the CNLI dataset: the verifier's high confidence indicates a stronger performance of LLM over SLM. That is, the verifier is more likely to suggest routing queries that will not gain much from the LLM. In contrast, `AutoMix` with POMDP, informed by the validation set, identifies this and adapts by discerning the optimal verifier probability range for routing. This underscores the utility of the meta-verifier in addressing verifier shortcomings.

On further investigation, we find that despite using identical prompts (sourced from Shaham et al. (2023)), the LLAMA2-13B model never answers 'Entailment', whereas LLAMA2-70B never answers with 'Contradiction'. While our meta-verifier doesn't directly process the LLAMA2-generated answers, it learns from the validation set that higher verifier confidence often corresponds to the true answer being 'Entailment', leading to a preference for LLM routing.

When does `AutoMix` not work? Analyzing the relatively poor performance of all methods on QUALITY, we find a substantial distribution shift

Method	CNLI			Quality			QASPER			NarrativeQA			COQA		
	C	P	Δ_{IBC}	C	P	Δ_{IBC}	C	P	Δ_{IBC}	C	P	Δ_{IBC}	C	P	Δ_{IBC}
SLM	1	40.1	-	1	47.5	-	1	14.0	-	1	20.3	-	1	48.1	-
FrugalGPT	37.4	59.2	66.1	49.7	66.5	**-2.5**	49.3	27.7	-1.1	45.9	26.0	2.5	30.3	57.1	13.1
`AutoMix` w/ SC	47.5	52.3	−17.0	15.2	52.8	−7.0	44.3	26.8	2.3	23.0	23.3	9.2	16.6	54.7	**55.5**
`AutoMix` w/ T	51.9	55.6	−3.5	37.7	61.6	−4.4	47.2	27.7	3.7	16.6	22.4	**12.2**	7.2	50.7	43.2
`AutoMix` w/ P	6.7	43.5	**88.7**	15.8	52.9	−11.8	45.2	27.6	**6.9**	9.9	21.4	6.4	6.9	50.5	43.7
LLM	50	55.5	-	50	67.1	-	50	28.1	-	50	26.4	-	50	61.4	-

Table 1: **Main Results:** highlighting the trade-offs between Cost (C), Performance (P), and Incremental Benefit per Cost (Δ_{IBC}) across various methods and datasets. The acronyms represent: SLM - Small Language Model, LLM- Large Language Model, `AutoMix` + T and `AutoMix` + P - variations of our proposed method with thresholding (T) and POMDP (P) based meta-verifiers, respectively. **`AutoMix` + POMDP** demonstrates a robust and consistent Δ_{IBC} across CNLI QASPER, NARRATIVE-QA, and COQA datasets, implying a judicious utilization of computational resources. **`AutoMix`** outperforms FrugalGPT across all datasets, despite latter having access to domain specific training and a near 0-cost verifier. While on CNLI **`AutoMix` + POMDP** provides a lift of around 90%, on QUALITY no variant of `AutoMix` or baseline works, a result we analyze in detail in Section 5.1.

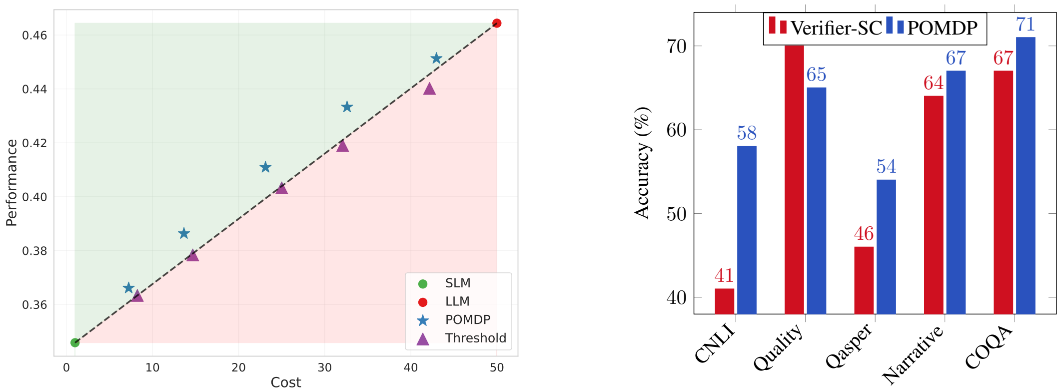

Figure 4: **Left:** Aggregated performance vs. cost for different methods on the small and large LLAMA2-13/70B. POMDP based meta-verifier is consistenly in the green region, signifying a higher Incremental Benefit per Cost (IBC). **Right:** The accuracy of the meta-verifier for both POMDP and Verifier-Self-Consistency (Verifier-SC) approaches across various datasets. Across all scenarios, the POMDP method consistently wins with up to 42% relative performance gains.

between the training and testing splits for the QUALITY dataset in Figure 5. Consequently, `AutoMix` +POMDP overfits a policy on the training set, which fails to generalize to the test set, resulting in inferior performance to `AutoMix` +SC. Further, neither variants of our model nor the baselines exhibit a positive Δ_{IBC} for the QUALITY dataset. This is attributed to the lack of correlation between $\Delta P_{LLM-SLM}$ and the verifier probability (Pearson coefficient = -0.03), implying that the verifier provides no valuable signal. In this context, the self-verifier's performance is almost equivalent to a random guess, and the meta-verifier also fails.

5.2 Key findings and takeaway

AutoMix is Effective in Low-Resource Scenarios Figure 7 demonstrates the performance dynamics of `AutoMix` and FrugalGPT with varying validation sizes. Notably, our method significantly outperforms FrugalGPT with limited data (under 2000 samples), despite the latter's domain-specific training and zero verifier cost. However, as training data increases, FrugalGPT narrows the performance gap by leveraging domain-specific training. This pattern indicates that `AutoMix` provides a particularly advantageous solution in real-world scenarios where data may be scarce.

Effectiveness of Few-shot Self-Verification In Appendix A.1, we evaluate few-shot self-verification quantitatively and qualitatively. We observe that the self-verification can effectively use context to identify errors in answers generated by SLM in many cases.

Improving Self-Verification with Task-Specific Prompt Engineering We explore the impact

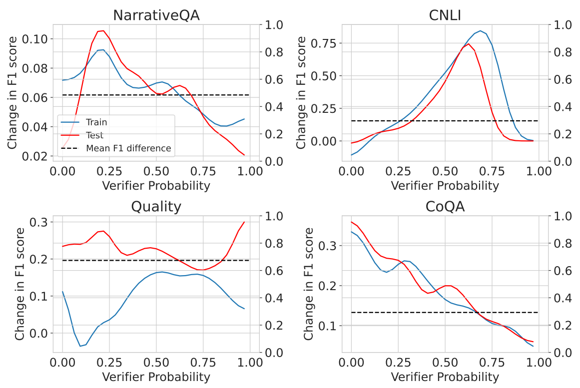

Figure 5: Delta improvements in F1-score of LLM over SLM for different verifier probabilities. A perfect verifier should be a line with negative slope: high delta when verifier confidence is low, and low delta when confidence is high. NARRATIVE-QA and COQA exhibit near perfect behavior. The trend is reversed for CNLI, with high confidence implying high delta. Unlike others, QUALITY shows no correlation between train and test splits, explaining poor lifts in learning based methods.

of task-specific prompt engineering on self-verification performance in Appendix A.2. While prompt engineering improves verifier accuracy, our meta-verifier remains robust in various settings and can beneficially leverage even a weak verifier.

6 Related Work

Self-Verification `AutoMix` aligns in spirit with works that aim to perform self-verification for reasoning problems, such as Weng et al. (2023); Jiang et al. (2023) (see Pan et al. (2023a) for a survey of recent self-verification and correction approaches). However, `AutoMix` uniquely harnesses context for verification instead of relying on LLM's knowledge (Dhuliawala et al., 2023) which can be challenging for reasoning problems (Madaan et al., 2023; Huang et al., 2023), and introduces a meta-verifier mechanism to offset the verifier's potential noise. Further, unlike Madaan et al. (2022), who utilize a corpus of past mistakes to gauge the likelihood of a model error for a new question, `AutoMix` uniquely utlizes context for verification. Finally, different from works that rely on external knowledge bases for verifying the outputs of language models (Peng et al., 2023; Gao et al., 2023; Pan et al., 2023b), `AutoMix` uses the context supplied with the question to verify the answer.

Our meta-verification approach can also be seen in the context of conformal prediction (Angelopoulos et al., 2023; Vovk et al., 2005) for a more robust self-verification. Ren et al. (2023) tie meta-verification more closely with conformal predictions for robot navigation, showing that layering predictions from a language model with a secondary mechanism helps in identifying situations that do not have adequate information for action.

Mixing Models Distinct from related work optimizing LLM inference cost by model switching and external verifiers (Chen et al., 2023; Zhu et al., 2023; vSakota et al., 2023), `AutoMix` obviates the need for verifier training through few-shot SLM model prompting and does not require upfront access to all input queries. When needed, the meta-verifier learned with only as few as 200 samples outperforms training specialized models. Our work is thus aligned with recent work that aims at composing different models and external tools for inference time improvement of language models (Khattab et al., 2023; Press et al., 2022; Yao et al., 2022; Zhou et al., 2022).

Adaptive Computation In contrast to adaptive computation and model routing methods that preempt computation via intermediate representations (Liu et al., 2020; Zhou et al., 2020; Schuster et al., 2021; Geng et al., 2021; Schuster et al., 2022; Madaan and Yang, 2022), `AutoMix` necessitates no architectural modifications and assumes only black-box access to APIs. Further, unlike AdaptiveConsistency (Aggarwal et al., 2023), which optimizes inference within a single LLM model, `AutoMix` flexibly optimizes between two models and transcends its utility in Self-Consistency.

7 Conclusion

`AutoMix` integrates black-box large language model (LLM) APIs into a multi-step problem-solving framework, optimizing the computational cost and performance trade-offs. `AutoMix` opens avenues for several interesting research directions. First, while self-verification and correction are challenging for LLMs in general, we find promising results using context-grounded few-shot verification, indicating that similar approaches may yield gain in other scenarios. Secondly, our work interweaves Good Old-Fashioned Artificial Intelligence (GOFAI) approaches with LLMs, demonstrating that the incorporation of a POMDP can boost the accuracy of a noisy few-shot verifier, showing the promise of this paradigm as an approach for improving LLMs during inference.

8 Limitations

While our empirical evidence demonstrates effectiveness, the broader applicability of `AutoMix` may vary depending on the specific models and datasets used. Further, `AutoMix` assumes a context-grounded reasoning setup for effective self-verification, which excludes tasks like factual question-answering and commonsense reasoning. Finally, as open-source models get powerful and inference costs decrease, serving a strong model for all queries might be feasible. However, there are still likely going to be latency and availability tradeoffs that might be handled using `AutoMix`.

References

Pranjal Aggarwal, Aman Madaan, Yiming Yang, and Mausam. 2023. Let's sample step by step: Adaptive-consistency for efficient reasoning with llms. *ArXiv*, abs/2305.11860.

Anastasios N Angelopoulos, Stephen Bates, et al. 2023. Conformal prediction: A gentle introduction. *Foundations and Trends® in Machine Learning*, 16(4):494–591.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901, Online. Curran Associates, Inc.

Lingjiao Chen, Matei A. Zaharia, and James Y. Zou. 2023. Frugalgpt: How to use large language models while reducing cost and improving performance. *ArXiv*, abs/2305.05176.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine learning challenges workshop*, pages 177–190. Springer.

Ido Dagan, Dan Roth, Fabio Zanzotto, and Mark Sammons. 2022. *Recognizing textual entailment: Models and applications*. Springer Nature.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4599–4610.

Mostafa Dehghani, Yi Tay, Anurag Arnab, Lucas Beyer, and Ashish Vaswani. 2021. The efficiency misnomer. In *International Conference on Learning Representations*.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2023. Chain-of-verification reduces hallucination in large language models. *arXiv preprint arXiv:2309.11495*.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jian, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D Hwang, et al. 2023. Faith and fate: Limits of transformers on compositionality. *arXiv preprint arXiv:2305.18654*.

Luyu Gao, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Arun Tejasvi Chaganty, Yicheng Fan, Vincent Zhao, Ni Lao, Hongrae Lee, Da-Cheng Juan, et al. 2023. Rarr: Researching and revising what language models say, using language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16477–16508.

Shijie Geng, Peng Gao, Zuohui Fu, and Yongfeng Zhang. 2021. Romebert: Robust training of multi-exit bert. *arXiv preprint arXiv:2101.09755*.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2023. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*.

Weisen Jiang, Han Shi, Longhui Yu, Zhengying Liu, Yu Zhang, Zhenguo Li, and James T Kwok. 2023. Backward reasoning in large language models for verification. *arXiv preprint arXiv:2308.07758*.

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

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Yuta Koreeda and Christopher D Manning. 2021. Contractnli: A dataset for document-level natural language inference for contracts. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1907–1919.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E.

Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Teven Le Scao and Alexander M Rush. 2021. How Many Data Points is a Prompt Worth? In *NAACL*.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021a. What Makes Good In-Context Examples for GPT-3? *arXiv:2101.06804 [cs]*. ArXiv: 2101.06804.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021b. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *arXiv preprint arXiv:2107.13586*.

Weijie Liu, Peng Zhou, Zhiruo Wang, Zhe Zhao, Haotang Deng, and Qi Ju. 2020. Fastbert: a self-distilling bert with adaptive inference time. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6035–6044.

Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. 2022. Memory-assisted prompt editing to improve GPT-3 after deployment. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2833–2861, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2023. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*.

Aman Madaan and Yiming Yang. 2022. Flowgen: Fast and slow graph generation. *arXiv preprint arXiv:2207.07656*.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, Yejin Choi, and Hannaneh Hajishirzi. 2021. Reframing Instructional Prompts to GPTk's Language. *arXiv preprint arXiv:2109.07830*.

George E. Monahan. 1982. State of the art—a survey of partially observable markov decision processes: Theory, models, and algorithms. *Management Science*, 28:1–16.

Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. 2023a. Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies. *arXiv preprint arXiv:2308.03188*.

Liangming Pan, Xiaobao Wu, Xinyuan Lu, Anh Tuan Luu, William Yang Wang, Min-Yen Kan, and Preslav Nakov. 2023b. Fact-checking complex claims with program-guided reasoning. *arXiv preprint arXiv:2305.12744*.

Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, et al. 2022. Quality: Question answering with long input texts, yes! In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5336–5358.

Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, et al. 2023. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *arXiv preprint arXiv:2302.12813*.

Adam Poliak. 2020. A survey on recognizing textual entailment as an nlp evaluation. *arXiv preprint arXiv:2010.03061*.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2022. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2019. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

Machel Reid and Graham Neubig. 2022. Learning to model editing processes. *arXiv preprint arXiv:2205.12374*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Allen Z Ren, Anushri Dixit, Alexandra Bodrova, Sumeet Singh, Stephen Tu, Noah Brown, Peng Xu, Leila Takayama, Fei Xia, Jake Varley, et al. 2023. Robots that ask for help: Uncertainty alignment for large language model planners. *arXiv preprint arXiv:2307.01928*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Timo Schick, Jane Dwivedi-Yu, Zhengbao Jiang, Fabio Petroni, Patrick Lewis, Gautier Izacard, Qingfei You, Christoforos Nalmpantis, Edouard Grave, and Sebastian Riedel. 2022. Peer: A collaborative language model.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q Tran, Yi Tay, and Donald Metzler. 2022. Confident adaptive language modeling. *arXiv preprint arXiv:2207.07061*.

Tal Schuster, Adam Fisch, Tommi Jaakkola, and Regina Barzilay. 2021. Consistent accelerated inference via confident adaptive transformers. In *Proceedings of* *the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4962–4979.

Uri Shaham, Maor Ivgi, Avia Efrat, Jonathan Berant, and Omer Levy. 2023. Zeroscrolls: A zero-shot benchmark for long text understanding.

Uri Shaham, Elad Segal, Maor Ivgi, Avia Efrat, Ori Yoran, Adi Haviv, Ankit Gupta, Wenhan Xiong, Mor Geva, Jonathan Berant, and Omer Levy. 2022. SCROLLS: Standardized CompaRison over long language sequences. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 12007–12021, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Noah Shinn, Beck Labash, and Ashwin Gopinath. 2023. Reflexion: an autonomous agent with dynamic memory and self-reflection.

Trey Smith and Reid Simmons. 2006. Focused real-time dynamic programming for mdps: Squeezing more out of a heuristic. In *AAAI*, pages 1227–1232.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. 2005. *Algorithmic learning in a random world*, volume 29. Springer.

Marija vSakota, Maxime Peyrard, and Robert West. 2023. Fly-swat or cannon? cost-effective language model choice via meta-modeling. *ArXiv*, abs/2308.06077.

Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. 2022. Generating sequences by learning to self-correct. *arXiv preprint arXiv:2211.00053*.

Yixuan Weng, Minjun Zhu, Shizhu He, Kang Liu, and Jun Zhao. 2022. Large language models are reasoners with self-verification. *arXiv preprint arXiv:2212.09561*.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Kang Liu, and Jun Zhao. 2023. Large language models are better reasoners with self-verification. *CoRR, abs/2212.09561*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Shuyan Zhou, Uri Alon, Frank F Xu, Zhengbao Jiang, and Graham Neubig. 2022. Docprompting: Generating code by retrieving the docs. In *The Eleventh International Conference on Learning Representations*.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. 2020. Bert loses patience: Fast and robust inference with early exit. *Advances in Neural Information Processing Systems*, 33:18330–18341.

Banghua Zhu, Ying Sheng, Lianmin Zheng, Clark W. Barrett, Michael I. Jordan, and Jiantao Jiao. 2023. On optimal caching and model multiplexing for large model inference. *ArXiv*, abs/2306.02003.

A Verifier Qualitative Analysis

A.1 How effective is few-shot self-verification?

One notable contribution of this work is the concept of few-shot self-verification of outputs. Self-Verification, especially for reasoning problems, poses its own set of challenges; however, our setup has a unique advantage: the capacity to utilize context to validate answers. For instance, the model can identify factual inaccuracies in the answer or discern apparent contradictions that might not have been evident during the initial response. But does this advantage translate to effective self-verification in practice? As depicted in Figure 6, aside from the CNLI dataset, few-shot self-verification succeeds in accurately identifying correct examples by assigning them higher probabilities across all other datasets.

Qualitative Analysis Representative Examples from our qualitative analysis are shown in Tables 2, 3, and 4.

FrugalGPT vs. `AutoMix` at different levels of data availability Figure 7 demonstrates the performance dynamics of `AutoMix` and FrugalGPT with varying validation data sizes. Notably, our method significantly outperforms FrugalGPT with limited data (under 2000 samples), despite the latter's domain-specific training and zero verifier cost. However, as training data increases, FrugalGPT narrows the performance gap by leveraging its domain-specific training. This pattern indicates that `AutoMix` provides a particularly advantageous solution in real-world scenarios where data may be scarce.

A.2 Domain-specific vs. Domain independent verifier

We used a single verifier with the LLAMA2-13B model to help steer the model. To avoid excessive prompt engineering, we used a generic prompt for all datasets. However, task-specific prompts generally help (Le Scao and Rush, 2021; Liu et al., 2021b; Mishra et al., 2021; Liu et al., 2021a). To investigate this, we create task specific prompts for CNLI by giving examples from legal domain in the prompt.

Figure 8 underscores the efficacy of employing task-specific verification prompts, ensuring a heightened probability allocation for accurate examples during the verification process. Interestingly, the enhanced verifier accuracy does not always directly translate to proportionate improvements in our proposed method, `AutoMix`, as evidenced in Table 5. This phenomenon higlights the role of meta-verifiers, adeptly negotiating through the outputs of potentially unreliable verifiers.

A.3 Meta-Verification Analysis

In Section 5.1, we discussed importance of meta-verifier for different datasets. Here we also discuss the case for QASPER dataset. Apart from standard QA, QASPER also requires models to identify questions that are unanswerable from the given context. However, when a SLM outputs 'Unanswerable', it is possible, that it is not able to figure out the answer from context, instead of the question actually being unanswerable. Therefore, we route all such queries to the LLM without consulting a verifier. Figure 9 shows the $\Delta P_{LLM-SLM}$vs verifier probability for QASPER. Interestingly, train and test show strong correlation for all except lower confidence thresholds. Further, the figure shows routing unanswerable queries directly to LLM is useful as it results in higher than average F1 gain.

B Methodology

B.1 POMDP

The Partially Observable Markov Decision Process (POMDP) presents a robust framework for handling decision-making problems under uncertainty, offering a structured way to manage and navigate through the decision spaces where the system's state is not fully observable (Monahan, 1982). A POMDP is defined by a tuple (S, A, T, R, Ω, O), where S is a set of states, A is a set of actions, T represents the state transition probabilities, R is the reward function, Ω is a set of observations, and O is the observation function.

In the context of meta-verifier, the *unobservable* states (S) represent the potential correctness of the verifier's predictions, categorized as *Simple*, *Complex*, and *Insolvable*. Actions (A) are binary: trust the verifier or invoke the LLM. The reward function (R) quantifies the cost or gain of making a particular action in a given state, steering the decision policy towards cost-effective actions. Observations (Ω) in our model are the verifier's probability outputs, discretized into bins. Specifically, we generate k=8 samples from the verifier, discretizing our observation space in intervals of size 0.125 ranging from 0 to 1.

The observation function (O) depicts the likeli-

Attribute	Content
Context	Which country grows the most tea? ... Shen sipped it. He liked it. He drank it all. Shen was proud of his new drink. *Truncated for brevity*
Question	When did he take it?
Generated Answer	He took it in 1990.
Gold Answer	about 2737 B.C
Verifier Output	The context does not mention that he took tea in 1990. Verification Decision: The AI generated answer is Incorrect.

Table 2: An example where the generated answer is verifiably incorrect, citing an unsupported date (1990), and the verifier successfully catches this discrepancy.

Attribute	Content
Context	Fort Lauderdale, Florida (CNN) – Just taking a sip of water or walking to the bathroom is excruciatingly painful for 15-year-old Michael Brewer, ... He suffered second- and third-degree burns over about two-thirds of his body, according to the hospital's associate director, Dr. Carl Schulman. *Truncated for brevity*
Question	How long before he recovers from his injury?
Generated Answer	he will never recover from his injury.
Gold Answer	He has a lifelong recovery.
Verifier Output	The context clearly states that the teen is recovering from his injuries. Verification Decision: The AI generated answer is Incorrect.

Table 3: The example demonstrates a challenging case for the verifier. The generated answer is a rephrasing of what's mentioned in the context, however the verifier fails to catch it.

Attribute	Content
Context	Pruning keeps the plant only three or four feet tall. This is an easy height for tea picking. Green tea is steamed right after the leaves are picked. Green tea _ its healthy qualities. For example, it may prevent heart disease. How did we get tea bag? The answer: by accident. *Truncated*
Question	What did they do to green tea after picking it?
SLM Answer	steamed.
Gold Answer	prune it
Verifier Output	The context mentions that green tea is steamed after being picked. Verification Decision: The AI generated answer is Correct.

Table 4: An instance where the verifier deems the answer correct, and the gold label was incorrect. The verifier is able to correctly infer that the context mentions tea being steamed after picking.

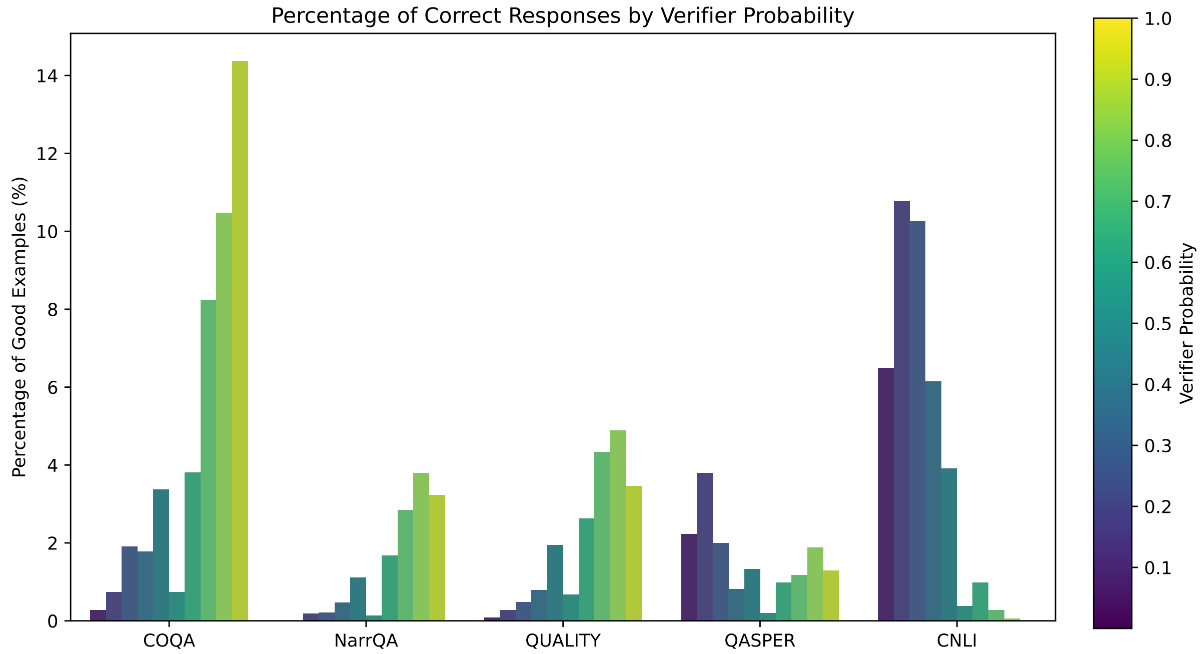

Figure 6: **Verifier Probability and Correctness:** Percentage of correct responses across distinct verifier probability bins, representing $P(\mathcal{C} = 1 \mid A_{\text{SLM}}, \mathcal{C}, q)$, where A_{SLM} is the answer from the Small Language Model, $\mathcal{C}$ is the context, and q is the query. Each bin represents a range of verifier probabilities and the corresponding accuracy of the responses within that probability range across various datasets. Notably, for all datasets, excluding CNLI and QASPER, a higher verification score generally corresponds to a larger proportion of correct examples, indicating that the verifier is, to an extent, capable of discerning the reliability of responses generated by itself. We use a meta-verifier to get around these noisy predictions.

	CNLI			CNLI-CV		
Method	Cost	Perf.	IBC_Lift	Cost	Perf.	IBC_Lift
SLM	1	40.1	-	1	40.1	-
FrugalGPT	37.4	59.2	**66.1**	37.4	59.2	**66.1**
Self-Consistency	47.5	52.3	-17.0	40.5	50.6	-15.5
`AutoMix`-Threshold	51.9	55.6	-3.5	28.1	46.9	-49.1
`AutoMix`-POMDP	6.7	43.5	88.7	15.8	45.2	12.4
LLM	50	55.5	-	50	55.5	-

Table 5: Despite the boost in verifier accuracy with task-specific prompts (Figure 8), `AutoMix` may not always benefit, highlighting the utility of even weak verifiers when supported by meta-verifiers.

hood of observing an observation given an action was taken and the system transitioned to a particular state. Using an appropriate observation function is crucial for POMDP to work. Specifically, we define observations probabilities in three ways:

- **1. Functional Form:** For each of the states s, the observation function O is defined as $O(s, v) = \frac{1}{K} \cdot v^{\gamma_s}$, where v is the verifier probability and $\gamma_s \in [0, \infty]$ is a hyperparameter for every state and K is normalizing factor. Intutively, a value of γ close to 1 indicates ideal calibration, with verifier probability v indicating true probability of being in a particular state. The values of γ_s's for the three states are determined based on the respective POMDP's performance on validation set based on the IBC-Lift.

- **2. Discrete Form:** An alternate option is to directly learn observation function O from the statistics of validation set. Since in validation set, we have access to the true state along with verifier probabilites of individual data instances, we can model observation function as $O(s, v) = \frac{\Sigma_{i=0}^{N} \mathbf{1}\{s_i = s \text{ and } v_i = v\}}{\Sigma_{i=0}^{N} \mathbf{1}\{s_i = s\}}$. The method has the advantage of being hyperparameter free and provides more accurate representation by computing the true observation probabilities on validation set. However, it performs worse than functional form, when either certain values of v or s are not well represented in validation set or in cases of high distribution shift between validation and test set.

- **3. Continous Form:** The continous form of POMDP follows the same formulation as in Discrete Form, except the fact the state space is represented by a tuple of SLM & LLM performance. Specifi-

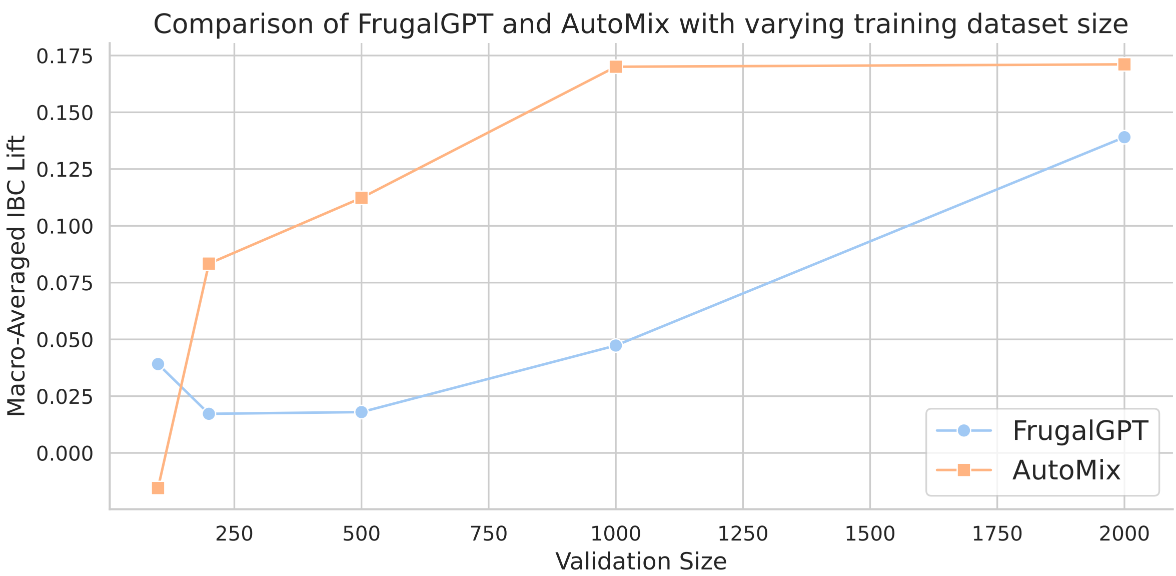

Figure 7: Comparison of `AutoMix` with FrugalGPT over varying Training Dataset Size. Despite zero-cost verifier and domain-specific training, `FrugalGPT` underperforms `AutoMix`. `AutoMix` is especially useful for limited data settings, with higher gains visible when dataset size is less than 1000.

cally, state space is represented by $\mathcal{S} = \{(P_{SLM}, P_{LLM}) | P_{SLM}, PLLM \in [0, 1]\}$, where P represents the performance of corresponding model on particular question. Since the performance (eg: F1 score) can be continous values, while we have discrete data (performance on individual scores), we apply gaussian smoothing (with standard deviation 1) followed by linear interpolation, to get observation probabilities for this continous state space.

Since both these methods have their strengths, and are independent of each other, we choose the best performing method on validation set.

This POMDP mechanism allows for optimal decision-making under uncertainty, balancing the cost and reliability of invoking the LLM. Through employing standard POMDP solving algorithms such as Focused Real-Time Dynamic Programming[3] (Smith and Simmons, 2006), we derive a policy that maps belief states (probability distributions over S) to actions. During inference, the learned policy effectively decides whether to trust the verifier's output or to invoke the LLM based on a combination of expected future rewards and computational costs.

Another advantage of the POMDP-based meta-verifier is its interpretability and customizability via reward assignment. For instance, in a "Needy" state, assigning a reward of +50 for invoking the LLM indicates a preference for accurate solutions over computational cost. Conversely, in a "Good" state, designating a reward of -10 for trusting the SLM encourages computational savings. This enables users to strategically balance solution quality against computational expenses, aligning with specific application needs.

[3] We use zmdp package `https://github.com/trey0/zmdp` for solving POMDP

B.2 Integrating Three Models with `AutoMix`

While the fundamental approach remains consistent, the three-model scenario diverges from its two-model counterpart in two key aspects: 1) the definition of observation probabilities, and 2) the evaluation methodology.

We employ a formulation akin to the continuous form of POMDP, as described in the previous section. However, in contrast to the two-model scenario, the observations can now fall into two categories: a) SLM verifier outputs on SLM answer, and b) SLM verifier outputs on SLM answer combined with MLM verifier outputs on MLM answer. The second category allows us to model more nuanced cues regarding the impact of verifiers on the final performance improvement. For instance, Figure 11 illustrates that when both verification probabilities are available, high $\delta_{MLM-LLM}F1$ regions can be detected, which is not feasible with a single verifier. This implies that the POMDP can

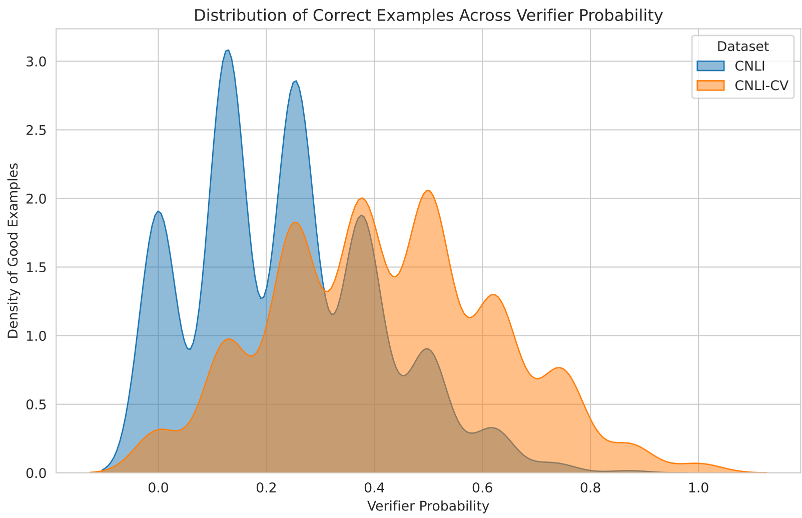

Figure 8: Enhancement of verifier accuracy using task-specific verification prompts, which allocate higher verification probabilities to more correct examples.

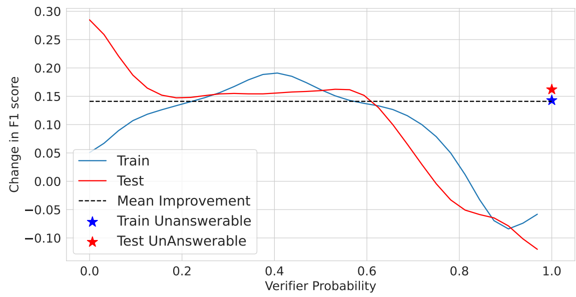

Figure 9: Delta imporvements in F1-score of LLM over SLM for different values of verifier probability for QASPER. While there are small regions where there is overall incremental benefit, routing queries where SLM outputs unanswerable or when verifier is confident is helpful.

make more informed decisions, an advantage that is empirically demonstrated in Results D.1.

In terms of evaluation, we consider two separate cases: 1) when the SLM-MLM-LLM curve is convex, and 2) when the curve is concave. In the convex case (as observed in the COQA dataset), it is advantageous to choose between the MLM and SLM in low-cost regions, while it is beneficial to choose between the MLM and LLM in high-cost regions. The suitable IBC curve is selected for evaluation accordingly. However, in the second case, when the IBC curves are concave, it would be more favorable to choose between the SLM and LLM, and completely ignore the MLM, as in terms of incremental performance per cost, it consistently presents a disadvantage. Thus, the $\text{IBC}_{SLM-LLM}$ is chosen for evaluation throughout. Although the evaluation presents two distinct cases, our $\texttt{AutoMix}_3$ framework is sufficiently general to identify instances where direct routing to LLM is needed even in the convex case, and also pinpoint cases where routing to MLM is beneficial in the concave scenario. This flexibility results in significantly superior performance.

C Expanding AutoMix to Three-Models

The preceding discussion focused on a two-model scenario involving the SLM and LLM. This section extends this framework to incorporate a third model, the MLM.

Our decision flow commences with the SLM generating an answer, which is then self-verified by the SLM. The verifier probability serves as an observation, guiding one of the following actions: 1) Reporting the SLM answer, 2) Running inference on the MLM or LLM and reporting the answer, or 3) Running inference on the MLM and verifying the answer. If action 3 is chosen, `AutoMix` has access to verification probabilities from both the SLM and MLM, which are used to decide whether to report the MLM's answer or switch to

```
# Meta-verifier POMDP File for narrative_qa

discount: 0.99
values: reward

# We have 6 states: 3 corresponding to the initial state before verifier is
    called, and 3 corresponding to the state after verifier is called
states: START_S START_C START_U SIMPLE COMPLEX UNSOLVABLE

# Effectively, we have 3 actions: 1.) The initial State where we run verifier
    2.) Report SLM's Answer 3.) Invoke LLM and Report its Answer
actions: Init Trust_SLM Invoke_LLM

# Observations lies in one of verifier probability bins. Eg: bin_correct_high
    represents Verifier outputs SLM answer as correct with high confidence
observations: bin_incorrect_low bin_incorrect_high bin_correct_low
    bin_correct_high

# Transition Model for Init action

T: Init
# Format: start_state : end_state : Transition_Probability

# Transition Model for Trust_SLM action
T: Trust_SLM
identity

# Transition Model for Invoke_LLM action
T: Invoke_LLM
identity

# Observation Model after "Init" action for narrative_qa
# Format: O : action : state : observation : probability

# Eaxmple: In SIMPLE cases, it is likely, SLM is correct and Verifier is
    Confident, while in UNSOLVABLE, SLM is incorrect (Lower Obs. Probability)
O : * : SIMPLE : bin_correct_high 0.8
O : * : COMPLEX : bin_correct_high 0.4
O : * : UNSOLVABLE : bin_correct_high 0.1

# Reward Model:
# Format: R: action : init_state : end_state : observation : probability

# Example: For COMPLEX state, Trusting SLM results in negative score, while
    invoking LLM results in a high +50 score.
R: Trust_SLM : COMPLEX : * : * -10
R: Invoke_LLM : COMPLEX : * : * +50
```

Figure 10: A sample POMDP specification file. POMDP requires defining states, actions, observations and relevant Transition, Observation Probabilities and Reward Values.

the LLM. Access to both the verifier probabilities provides AutoMix's meta-verifier with a richer observation signal. For instance, a neutral SLM verification signal combined with a neutral MLM verification signal will likely route the queries to the MLM. In comparison, an uncertain SLM verification signal and a neutral MLM verification signal will more likely be routed to LLM. In Section D.1, we compare different variants of AutoMix, highlighting the individual importance of each state in AutoMix's formulation. Further details are provided in Appendix B.2.

Meta-Verifier in the Three-Model Case We employ a similar POMDP formulation as in the two-model scenario but with a broader range of actions due to the inclusion of the third model. The states are now represented as a tuple of performance metrics for each of the three models. Formally, the state space is denoted as $\mathcal{S} = \{(P_{SLM}, P_{MLM}, P_{LLM}) | P_{SLM}, P_{MLM}, PLLM \in [0,1]\}$, where P denotes the performance of the respective model. For instance, if only the LLM can correctly solve the problem, the state will be represented as (0,0,1). AutoMix maintains

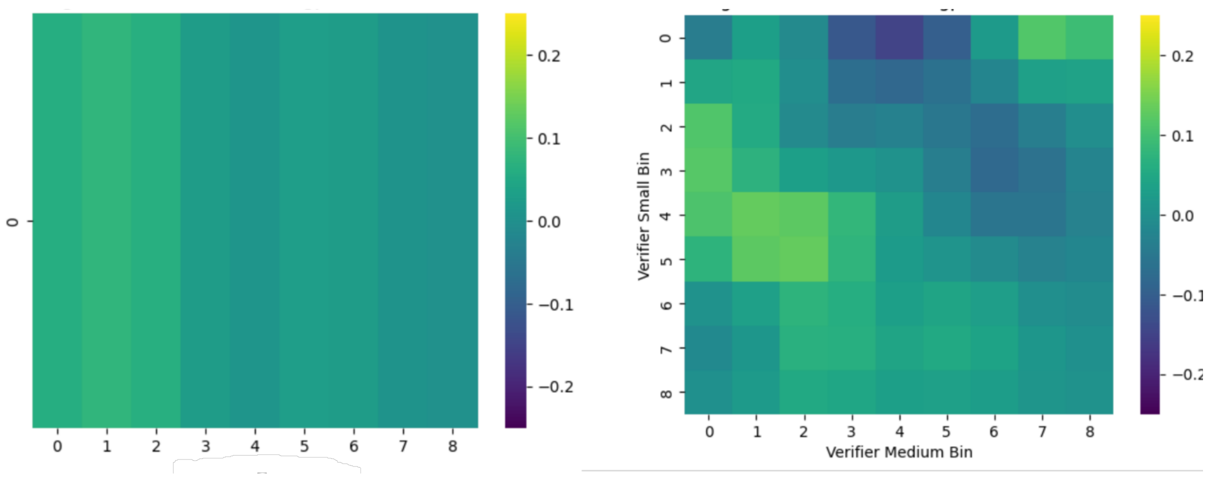

Figure 11: In the figure we compare delta improvement in F1 score from LLAMA2-70B to GPT-4 on COQA dataset, for different verifier probabilities. The graphs are smoothened using gaussian smoothing with standard deviation=1. On left, we vary only the MLM verifier, and on right we vary both SLM and MLM verifiers. The latter case provides much richer, thus showing importance of incorporating both of them in our $\texttt{AutoMix}_3$ formulation.

a belief over all possible states and updates this belief based on the verifier probabilities, which serve as observations. The model can observe either the SLM verifier probability or the SLM and MLM verifier probabilities. The observation probabilities are learned from the validation set as in the previous section.

D Additional Details on the Experimental Setup

For evaluation, we utilize the validation sets from Shaham et al. (2022) for NARRATIVE-QA, QASPER, CNLI, and QUALITY, along with their provided prompts. For COQA, we employ its validation split and adapt the QUALITY prompt. For consistency, 1000 instances are sampled from val set of each dataset. The procedure is repeated over 10 seeds, to reduce variance. Regardless of dataset, identical input prompts are dispatched to both SLM and potentially LLM, ensuring consistent input processing costs. The output length is fixed in multichoice datasets like CNLI and QUALITY, and the brevity of responses in other datasets allows us to assume uniform output processing costs. We use greedy decoding (temperature 0) and draw a single sample for both the SLM and LLM. For verification, we generate 8 samples per question (temperature = 1), which has negligible cost owing to large context. In analysis section 5.1, we draw 32 samples from verifier and apply gaussian smoothing to curves for convenient visualization without varying noise.

For running our experiments we use LLAMA2-13B and LLAMA2-70B models from huggingface[4]. We use vllm (Kwon et al., 2023) for hosting models for inference.

D.1 Results of Automix w/ 3 Models

In this section, we evaluate the performance of `AutoMix` when applied to a three-model scenario, as described in Section C. Specifically, we employ LLAMA2-13B as the SLM, LLAMA2-70B as the MLM, and GPT-4 as the LLM. Due to cost constraints, our evaluation is conducted on a subset of 1000 examples from the COQA dataset. The results of this evaluation are presented in Figure 12.

Our findings reveal that $\texttt{AutoMix}_3$ consistently outperforms the IBC curve for both the SLM-MLM and MLM-LLM cost regions. We also compare $\texttt{AutoMix}_3$ against a baseline, *Union* `AutoMix`, which chooses between the two-model variants $\texttt{AutoMix}_{SLM-MLM}$ and $\texttt{AutoMix}_{MLM-LLM}$, depending on the cost requirements specified by the end-user. For instance, if the desired average cost is less than that of the MLM, $\texttt{AutoMix}_{SLM-MLM}$ is employed, whereas $\texttt{AutoMix}_{MLM-LLM}$ is utilized for cost regions exceeding that of the MLM. $\texttt{AutoMix}_3$ outperforms the baseline consistently on all cost regions. This better performance can be attributed to the fact that $\texttt{AutoMix}_3$ has access to verifier probabilities from both LLAMA2-13B and LLAMA2-70B, which provides a richer signal to POMDP, resulting in taking

[4]Models available at: https://huggingface.co/meta-llama/Llama-2-13b-hf and https://huggingface.co/meta-llama/Llama-2-70b-hf

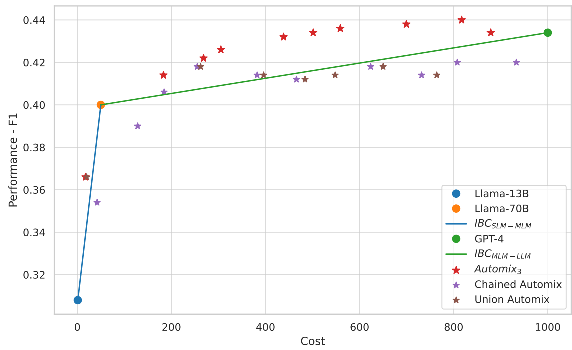

Figure 12: AutoMix with 3 models: LLAMA2-13B, LLAMA2-70B and GPT-4. AutoMix method shows consistent IBC lifts for both SLM-MLM and MLM-LLM regions. Further, compared to chaining two AutoMix models or using the union of two AutoMixes, AutoMix$_3$ provide significant improvements.

more informed actions. Further, we consider a baseline by chaining AutoMix$_{SLM-MLM}$ with AutoMix$_{MLM-LLM}$. The query first goes to the SLM, and an AutoMix$_{\text{SLM}-\text{MLM}}$ decides between reporting the SLM answer or routing to the MLM. In the latter's case, a second AutoMix$_{\text{MLM}-\text{LLM}}$ repeats the procedure using the MLM and LLM models. We call this method 'Chained AutoMix,' and it underperforms across the board. This is primarily because it cannot directly route queries from the SLM to the LLM. Additionally, whenever 'Chained AutoMix' prompts the MLM, it invariably uses the costly verifier, even in cases where it might not be necessary. This inefficient use of resources contributes to its subpar performance.

E Few-Shot Prompts

```
Story:
{relevant parts of the story}

{instruction}

Question: {question}

Answer:
```

Listing 2: **Task Prompt.** We experiment with long-context reasoning tasks, which require answering questions from stories, legal contracts, research papers, and novels.

E.1 Verifier Prompts

```
Context: {context}

Question: {question}

AI Generated Answer: {generated_answer}

Instruction: Your task is to evaluate
↪   if the AI Generated Answer is
↪   correct, based on the provided
↪   context and question. Provide the
↪   judgement and reasoning for each
↪   case. Choose between Correct or
↪   Incorrect.

Evaluation:"'
```

Listing 3: **Verification Prompt.** The verification process is framed as a natural language entailment task, where the model determines the validity of the model-generated answer with respect to the context and question.

```
Context: The manuscript, discovered in
↪ 1980 in a dusty attic, turned out
↪ to be a lost work of Shakespeare.

Question: Whose lost work was
↪ discovered in a dusty attic in
↪ 1980?

AI Generated Answer: Shakespeare

Instruction: Your task is to evaluate
↪ if the AI Generated Answer is
↪ correct, based on the provided
↪ context and question. Provide the
↪ judgement and reasoning for each
↪ case. Choose between Correct or
↪ Incorrect.

Evaluation: The context specifically
↪ mentions that a lost work of
↪ Shakespeare was discovered in 1980
↪ in a dusty attic.

Verification Decision: The AI generated
↪ answer is Correct.

---

Context: The celestial event, known as
↪ the Pink Moon, is unique to the
↪ month of April and has cultural
↪ significance in many indigenous
↪ tribes.

Question: In which month does the
↪ celestial event, the Pink Moon,
↪ occur?

AI Generated Answer: July

Instruction: Your task is to evaluate
↪ if the AI Generated Answer is
↪ correct, based on the provided
↪ context and question. Provide the
↪ judgement and reasoning for each
↪ case. Choose between Correct or
↪ Incorrect.

Evaluation: The context clearly states
↪ that the Pink Moon is unique to the
↪ month of April.

Verification Decision: The AI generated
↪ answer is Incorrect.

---

{truncated examples}

Context: {context}

Question: {question}

AI Generated Answer: {generated_answer}

Instruction: Your task is to evaluate
↪ if the AI Generated Answer is
↪ correct, based on the provided
↪ context and question. Provide the
↪ judgement and reasoning for each
↪ case. Choose between Correct or
↪ Incorrect.

Evaluation:
```


Listing 4: **Few-Shot Verifier Prompts:** 3-shot verifier