AutoMix: Mixing Models with Few-shot Self and Meta Verification

Anonymous Author(s)
Affiliation
Address
email

Abstract

Large language models (LLMs) are now available in various sizes and configurations from cloud API providers. 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 57%.

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., 2023; 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 singular model across all problem-solving stages, demonstrating effectiveness in certain scenarios (Madaan et al., 2023; Welleck et al., 2022). 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. This approach 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(Welleck et al., 2022; Reid and Neubig, 2022), which may not always be feasible as modern LLMs rely on access to black box APIs. To address these challenges, we propose a new method, which we call `AutoMix`. In contrast to existing approaches, `AutoMix` fully leverages black-box LLM APIs, avoiding the need for separate models or access to logits using few-shot learning (Brown et al., 2020) and meta-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 delineate tasks as Simple or Complex for model routing, `AutoMix` integrates a third category of *Unsolvable* queries. These queries are likely unsolvable even by a Large Language Model (LLM) and should not be routed to larger models if identified early enough. This consideration allows `AutoMix` to judiciously allocate computational resources, preventing unwarranted computational spending on these particularly challenging instances.

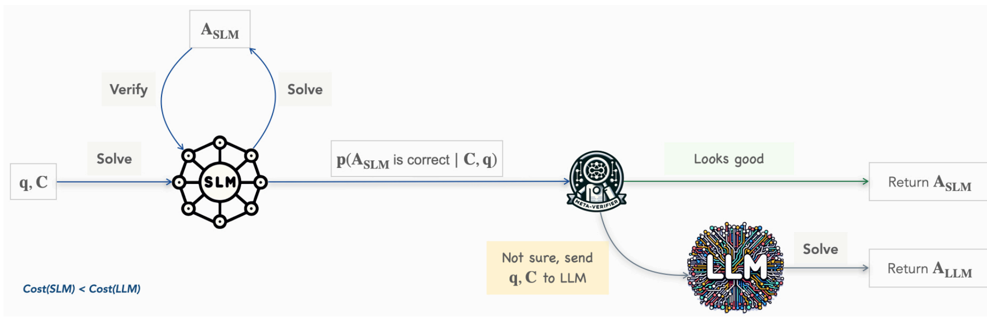

Figure 1: `AutoMix`: Given a context C and question q, an initial answer A_{SLM} is generated with the smaller language model (SLM). A_{SLM} is verified by the SLM, yielding a noisy verification score. Based on the meta-verifier's decision, either A_{SLM} is returned if deemed satisfactory, or the task is rerouted to a larger language model (LLM) to enhance accuracy. The looping arrow around the SLM symbolizes self-correction before proceeding to the meta-verifier.

We use context-grounded few-shot entailment to quantify the uncertainty in an answer's correctness (Poliak, 2020; Dagan et al., 2022). However, recognizing that verifications can sometimes be inconsistent or noisy, 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 albeit 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 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 SLM and LLM. Our results demonstrate that `AutoMix` surpasses established baselines, enhancing the incremental benefit per cost by up to 57%.

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 C.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 to generate an accurate and coherent answer, consistent with the provided context. We deploy two distinct models: a smaller, cost-efficient model, denoted as SLM, and a larger, more accurate yet costly model, LLM. Our objective is to optimize performance while staying economical. We use a verifier, $\mathcal{V}$, to ascertain the validity of SLM's outputs and decide if a query should be redirected to LLM. We start by generating an initial answer, $\mathcal{A}_s$ using the smaller SLM. Next, we need to assess the trustworthiness of $\mathcal{A}_s$. To this end, we use a few-shot verifier. Our choice of tasks is motivated by two key concerns. First, longer queries are more computationally demanding, underscoring the need for an approach like `AutoMix` to navigate the cost-accuracy trade-off. Second, Context allows the verifier to cross check the preliminary answers with available information, aiding in identifying inconsistencies as ungrounded is challenging (Pan et al., 2023; Huang et al., 2023).

Verification is framed as an entailment task (Poliak, 2020; Dagan et al., 2022). The objective is 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.

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 for the LLM is resource-inefficient without enhancing performance. While ascertaining the ground truth of query difficulty is non-trivial, verification probability and historical data 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 versatile meta-verifier can adopt various advanced learning strategies, from supervised to reinforcement learning, 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 $H(t) = 0$ for $t < 0$ and $H(t) = 1$ for $t \geq 0$. For black-box language models, the probability of correctness can be derived by sampling $k > 1$ samples at a higher sampling temperature.

Using a POMDP In the context of meta-verifier, we observe that all the queries in this two language model setup could be categorized in 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 can't be addressed by either LLM or SLM. Hence, a ground truth oracle should route only the complex queries but not unsolvable queries. Since the ground truth state is not known 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. See Appendix A.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 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 potential for extension to streaming settings for optimizing across multiple queries or a fixed time duration.

procedure ANSWERQUERY($\mathcal{C}, q$) ▷ Context, Question

  $\mathcal{A}_s \leftarrow$ solve(SLM, $\mathcal{C}, q$)
  $v \leftarrow$ self-verify($\mathcal{A}_s, \mathcal{C}, q$)
  if $m(v, \mathcal{A}_s, \mathcal{C}, q)$ **then** ▷ Check meta-verifier decision
    if $v \geq 0.5$ **then**
      return $\mathcal{A}_s$
    else
      return solve(LLM, $\mathcal{C}, q$)
    end if
  else
    return $\mathcal{A}_s$
  end if
end procedure

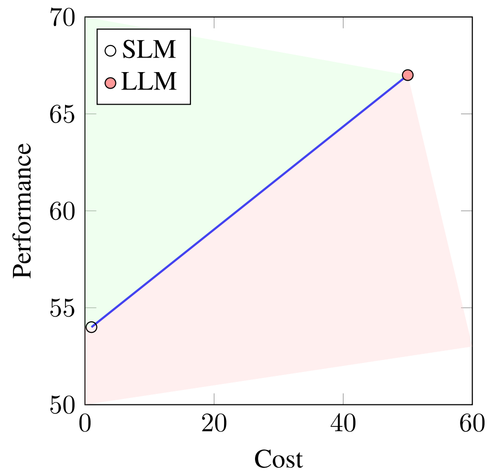

Figure 2: **Left:** `AutoMix` algorithm. **Right**: Performance vs. Cost curve. The slope between SLM and LLM provides a way to the Incremental Benefit per Cost (IBC) for methods that mix models. Methods with a steeper slope than this reference when plotted against SLM have a positive IBC (green region), whereas those below the reference have a negative IBC (red region), falling into the red region.

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 (1)).

$$\text{IBC}_M = \frac{P_M - P_{\text{SLM}}}{C_M - C_{\text{SLM}}}, \quad \text{IBC}_{\text{BASE}} = \frac{P_{\text{LLM}} - P_{\text{SLM}}}{C_{\text{LLM}} - C_{\text{SLM}}}, \quad \Delta_{\text{IBC}}(M) = \frac{\text{IBC}_M - \text{IBC}_{\text{BASE}}}{\text{IBC}_{\text{BASE}}} \times 100 \tag{1}$$

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 2)

Cost Calculation The total cost, C_M, for a method M utilizing the Small Language Model (SLM) for initial answer generation and verification, and the Large Language Model (LLM) as needed, is computed as: $C_M = 2 \cdot C_{\text{SLM}} + w_{\text{LLM}} \cdot C_{\text{LLM}}$. Here, $w_{\text{LLM}} \in [0, 1]$ represents the proportion of LLM usage, with values indicating exclusive ($w_{\text{LLM}} = 1$) or no usage ($w_{\text{LLM}} = 0$) of LLM. While we utilize SLM for verification, it is worth noting that a different verifier model could also be employed, which would alter the cost calculations accordingly.

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 [1].

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

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	29.8	-	1	14.0	-	1	20.3	-	1	48.1	-
FrugalGPT	37.4	59.2	**66.1**	49.7	42.0	-2.1	49.3	27.7	-1.1	45.9	26.0	2.5	30.3	57.1	13.1
SC	43.6	51.2	-17.3	11.8	32.3	-9.8	46.6	27.5	2.6	23.4	23.1	1.2	16.9	54.6	**49.6**
AutoMix +T	51.9	55.4	-4.5	24.8	36.1	3.5	38.4	24.9	-12.0	13.2	21.9	2.4	8.3	51.0	31.7
AutoMix +P	5.5	42.3	57.0	9.6	32.1	**4.0**	45.4	27.4	**5.0**	10.3	21.5	**3.6**	7.9	50.8	42.5
LLM	50	55.5	-	50	42.3	-	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 the **Quality**, QASPER, NARRATIVE-QA, and COQA datasets, implying a judicious utilization of computational resources.

Datasets We experiment with a diverse set of datasets: NARRATIVE-QA (Kočiskỳ et al., 2018) for full-length book and movie script QA, QASPER (Dasigi et al., 2021) for research paper QA, CNLI (Koreeda and Manning, 2021) for NLI tasks, QUALITY (Pang et al., 2022) for multiple-choice questions from long articles, and COQA (Reddy et al., 2019) for conversational comprehension QA. The datasets are evaluated on F1 score, accuracy, and exact match based on answer format.

Baselines Our baselines include: i) **Verifier Self-Consistency** (Wang et al., 2022), where we prompt SLM with our entailment verifier and draw 8 samples (temperature 0.7). The majority label routes the query to LLM or not. We cache the KV values for the (long) input prompt, so only a single forward pass is done. The cost of this verifier is the same as the cost of SLM. and ii) **Frugal GPT (F) (Chen et al., 2023)** 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 adhere to a low-resource setting, utilizing 1000 training examples per dataset.

Proposed approaches We experiment with two different types of meta-verifiers: threshold and POMDP-based. i) **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. ii) **AutoMix + POMDP**: This method optimizes routing decisions using a POMDP solver (Smith and Simmons, 2006), given verifier outputs and observation probabilities learned on the validation set (detailed in Appendix A.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 QUALITY,QASPER,NARRATIVE-QA,COQA, AutoMix beat FrugalGPT despite the latter having access to domain-specific training and low verifier cost. In Figure 3 (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 3 (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 33% demonstrating our proposed meta-verifier's importance in few-shot verification setups.

4.2 Key findings and takeaway

AutoMix is Effective in Low-Resource Scenarios Figure 6 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 perfor-

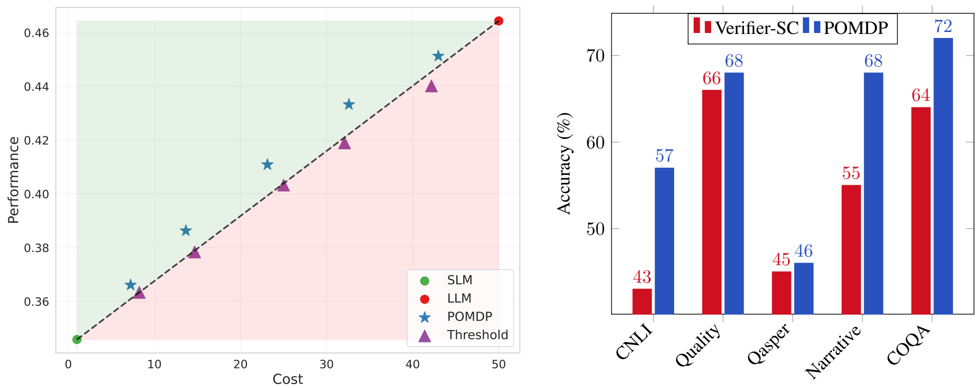

Figure 3: **Left:** Aggregated performance vs. cost for different methods on the small and large LLAMA2-13/70B. POMDP based meta-verifier is consistenly in the green region, signifying a higher Incremental Benefit per Cost (IBC). **Right:** The accuracy of the meta-verifier for both POMDP and Verifier-Self-Consistency (Verifier-SC) approaches across various datasets. Across all scenarios, the POMDP method consistently wins with up to 33% relative performance gains.

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

5 Related Work

1. 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. **2. Adaptive Computation** In contrast to adaptive computation 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), `AutoMix` necessitates no architectural modifications. 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. **3. Self-Verification** `AutoMix` aligns in spirit with works that aim to perform self-verification for reasoning problems, such as Weng et al. (2023); Pan et al. (2023). However, `AutoMix` uniquely harnesses context for verification instead of relying on LLM's knowledge, and introduces a meta-verifier mechanism to offset the verifier's potential noise.

6 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.

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.

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, 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.

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*.

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*.

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.

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, 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. 2023. Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies. *arXiv preprint arXiv:2308.03188*.

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.

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

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*.

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.

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*.

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

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Huai hsin Chi, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *ArXiv*, abs/2203.11171.

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, 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*.

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 Methodology

A.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 likelihood 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 two 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 $O(s, v) = \frac{\Sigma_{i=0}^{N} \mathbf{1}\{s_i = s \ and \ v_i = v\}}{\Sigma_{i=0}^{N} \mathbf{1}\{s_i = s\}}$. 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.

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[2] (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.

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

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

B Verifier Qualitative Analysis

B.1 How effective is few-shot verification?

One of the contributions of this work is few-shot self-verification of outputs. While Self-Verification is challenging in general for reasoning problems, our setup offers an advantage: the context allows us to check for the validity of the answer using the context. For example, the model can spot factual errors in the answer, or apparent contradictions that may not be clear when the question is answered for the first time. Are the models able to use this advantage for effective self-verification? Figure 5 shows that for all datasets except CNLI, few-shot self-verification effectively identifies correct example, assigning a higher probability to them.

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

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.

FrugalGPT vs. `AutoMix` at different levels of data availability Figure 6 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.

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 lable was incorrect. The verifier is able to correctly infer that the context mentions tea being steamed after picking.

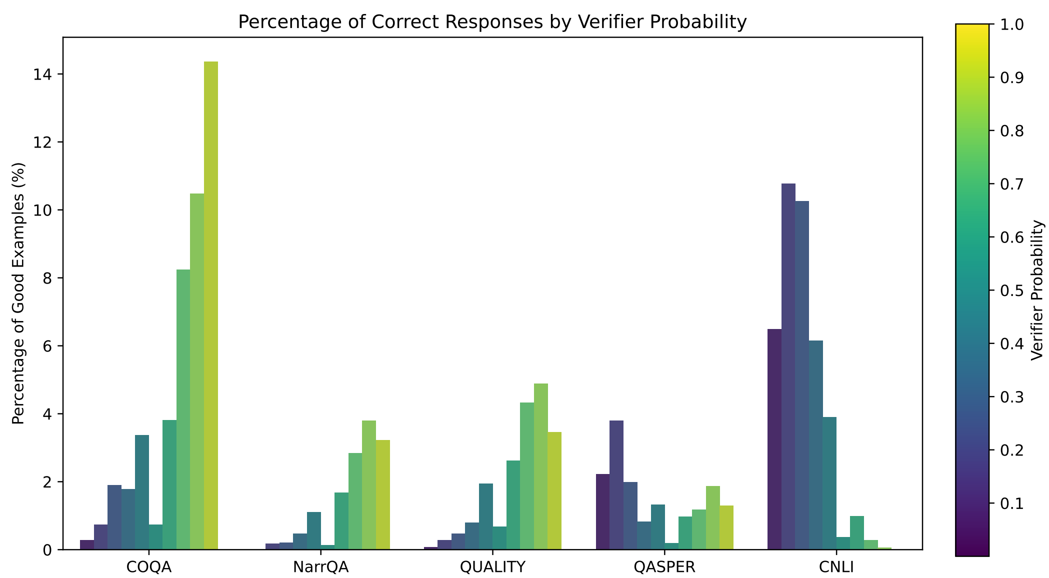

Figure 5: **Verifier Probability and Correctness:** Percentage of correct responses across distinct verifier probability bins, representing $P(\mathcal{C} = 1 \mid A_{\text{SLM}}, \mathcal{C}, q)$, where A_{SLM} is the answer from the Small Language Model, $\mathcal{C}$ is the context, and q is the query. 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.

B.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 7 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.

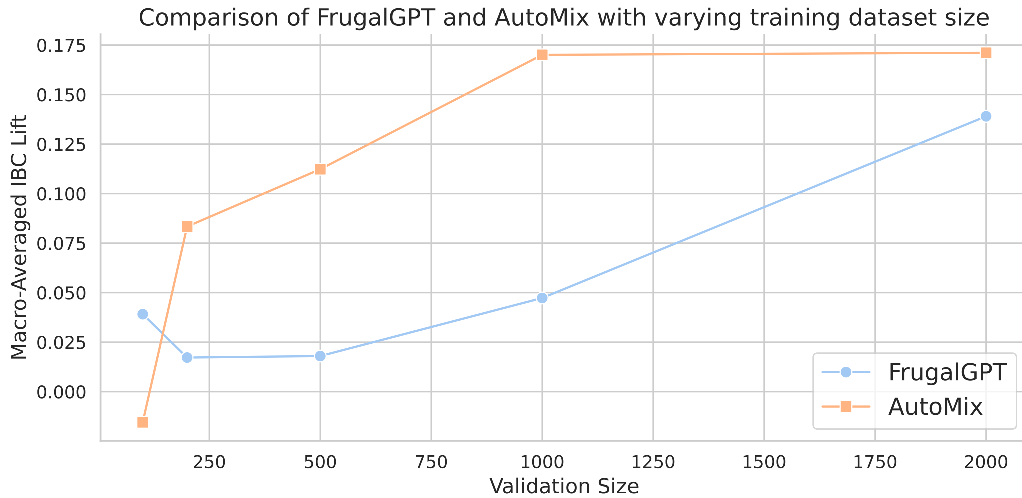

Figure 6: Comparison of `AutoMix` with FrugalGPT over varying Training Dataset Size. Despite zero-cost verifier and domain-specific training, `FrugalGPT` underperforms `AutoMix`. `AutoMix` is especially useful for limited data settings, with higher gains visible when dataset size is less than 1000.

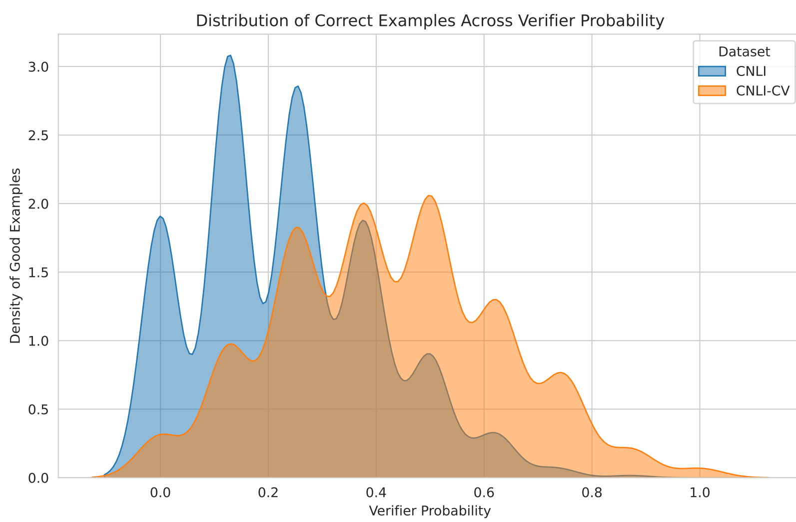

Figure 7: Enhancement of verifier accuracy using task-specific verification prompts, which allocate higher verification probabilities to more correct examples.

	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	43.6	51.2	-17.3	40.5	50.6	-15.5
`AutoMix-Threshold`	51.9	55.4	-4.5	28.1	46.9	-49.1
`AutoMix-POMDP`	5.5	42.3	57.0	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 7), `AutoMix` may not always benefit, highlighting the utility of even weak verifiers when supported by meta-verifiers.

C 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.

```
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.

C.1 Verifier Prompts

```
### NARRATIVE_QA

Story:
{context}

You are given a story, which can be either a novel or a movie script, and a
↪   question. Answer the question as concisely as you can, using a single phrase
↪   if possible.

Question: {question}

Answer: The answer is'""",
        "truncation_message": "... [The rest of the story is omitted]\n\n",

### QASPER

Article:
{context}

You are given a scientific article and a question. Answer the question as
↪   concisely as you can, using a single phrase or sentence if possible. If the
↪   question cannot be answered based on the information in the article, write
↪   'unanswerable'. If the question is a yes/no question, answer 'yes', 'no', or
↪   'unanswerable'.

Question: {question}

Answer: The answer is'""",
        "truncation_message": "... [The rest of the article is omitted]\n\n",

### QUALITY

Story:
{context}

You are provided a story and a multiple-choice question with 4 possible answers
↪   (marked by A, B, C, D). Choose the best answer by writing its corresponding
↪   letter (either A, B, C, or D).

Question and Possible Answers: {question}

Answer: The answer is'""",
        "truncation_message": "... [The rest of the story is omitted]\n\n",

### CNLI

Contract:
{context}

You are given a non-disclosure agreement and a sentence that proposes a hypothesis
↪   based on the agreement. Choose whether the hypothesis is entailed by the
↪   agreement, contradicted by the agreement, or not mentioned by (neutral to) the
↪   agreement. If the hypothesis is entailed by the agreement, write 'Entailment'.
↪   If the hypothesis is contradicted by the agreement, write 'Contradiction'. If
↪   the hypothesis is not mentioned by the agreement, write 'Not mentioned'.

Hypothesis: {question}

Answer: The answer is'""",
        "truncation_message": "... [The rest of the contract is omitted]\n\n",
}
```

Listing 4: **Few-Shot Prompts:** Dataset Specific Few-Shot Prompts used for SLM and LLM on NARRATIVE-QA, QASPER, QUALITY, CNLI. A general structure of context, instruction, question and answer is followed.

```
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 5: **Few-Shot Verifier Prompts:** 3-shot verifier prompt for evaluating the correctness of SLM's answer. The same prompt is used for all datasets.