Human-in-the-Loop through Chain-of-Thought

Anonymous ACL submission

Abstract

While the emergence of powerful language models along with Chain-of-thought prompting has made automation more and more omnipresent, it sometimes demonstrates its weakness in long-term or multi-step logical reasoning. For example, users don't always get desirable answers for complex mathematical problems without human involvement. Against this background, we present the **Manual Correction System (MCS)** — a human-in-the-loop system enhanced by Chain-of-Thought prompting, which explores how manual correction of sub-logics in rationales can improve LLM's reasoning performance. Moving one step forward, considering a system with human-in-the-loop involves more than having humans improve performance but also controlling the cost. Therefore, we post a **Cost-utility Analysis Model for Human-in-the-Loop systems (CAMLOP)** based on classical economics theory to analyze, quantify and balance the utility and the corresponding cost. We conduct experiments of MCS and CAMLOP with twelve datasets. A significant advantage w.r.t cost and utility proves its superiority over strong baselines.

1 Introduction

Large language model-based Artificial Intelligence systems are augmenting humans in certain roles, and soon this trend will expand to the vast majority of the workforce. However, while the emergence of powerful language models (Sanh et al., 2021; Ouyang et al., 2022; Zhang et al., 2022; Shao et al., 2023) has made automation omnipresent, it sometimes demonstrates its weakness in long-term or multi-step logical reasoning (Hosseini et al., 2014; Kushman et al., 2014; Koncel-Kedziorski et al., 2015; Roy and Roth, 2016). For example, users don't always get desirable answers for a mathematical problem without human involvement. To make tangible progress in mitigating these errors is where we need humans, and a system with human-in-the-loop involves more than having humans improve performance but also controlling the cost. Against this background, there comes a timing question: how to get a human-in-the-loop system in the most effective (namely, high-utility) and low-cost way?

See Fig. 1 as an example. For humans, solving the whole problem in the leftmost box is often more difficult than solving one of the sub-logics (*e.g.*, $2*(16-3) = 25$). Correction of the erroneous sub-logic (*e.g.*, $2*(16-3) = 25 \rightarrow 2*(16-3) = 26$) helps LLM reach a correct final answer.

In the last few years, thanks to explorations in Large Language Models (LLMs) and advances in in-context learning (ICL) technologies, giant breakthroughs have been obtained. Just by being fed an instruction, models can function very well on that task without manual finetuning (Brown et al., 2020a). This provides a chance for a human to change the predicted results via natural language instructions as a flexible and friendly interface. Furthermore, changing the rationale for chain-of-thought (CoT) prompting (Wei et al., 2022) is even more user-friendly since short and simple sub-logics in the rationale are easy for humans to handle. Whereas manual correction helps, the labor of this additional correction stage brings a direct and indirect cost (See Sec. 3 for more details). When and how humans intervene will greatly affect the cost and utility. Until recently, few researchers had explored this balance in ICL.

We present the **M**anual **C**orrection **S**ystem (MCS ; Sec. 2) — a human-in-the-loop system, which explores when and how manual correction of rationales can efficiently improve LLM's reasoning ability. To our knowledge, MCS is the first human-in-the-loop system leveraging rationales. As shown in Fig. 1, MCS consists of four stages: prompting the LLM with CoT, automatically filtering out the incorrectly predicted samples, human correcting their rationales, and prompting the LLM using CoT again to obtain the final answer. Referring

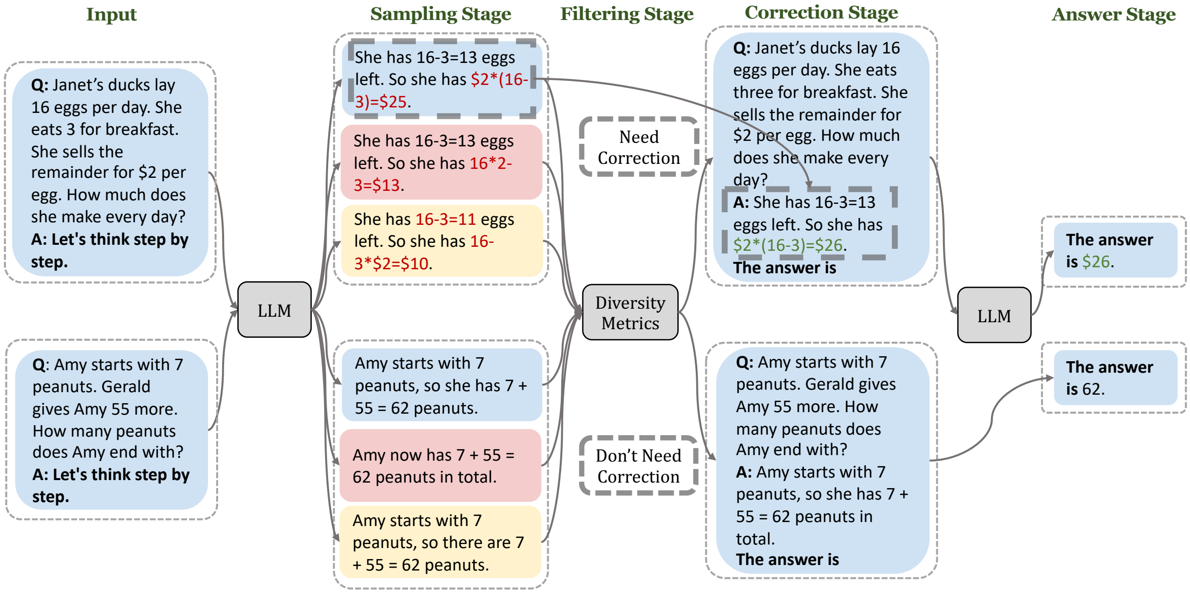

Figure 1: MCS comprises four stages: (1) **sampling stage** prompting the LLM using CoT prompting and replacing the greedy decoding by sampling from the LLM's decoder to generate a set of rationales (*i.e.*, the complete logical chain of CoT output); (2) **filtering stage** filtering out the samples ranked high by Diversity Entropy; (3) **correction stage** manually adding, deleting and modifying erroneous sub-logics in the most likely rationale of the filtered sample, and (4) **answer stage** prompting the LLM using CoT prompting again with manually corrected sub-logics and using greedy decoding to obtain the final answer.

to the "when" problem, we consider a diversity-based method to get a cue to indicate when humans should be involved, so as to reduce human labor as much as possible (See. 2.1). The diversity-based method is inspired by the diversity of the rationales. We have found that even when the desired answer is fixed, introducing the diversity degree of the rationales can be highly beneficial; therefore we introduce Diversity Metrics, as commonly used in Active Learning field (Brinker, 2003; Yang et al., 2015; Agarwal et al., 2020), to find data points requiring manual intervention. Then it comes to the "how" problem (See. 2.2). We empirically prove the viability of paying attention to sub-logics instead of the whole problem. We define three operations (*i.e.*, modifying, adding, and deleting) that a human can perform on the sub-logics of rationales for efficiency and simplification.

With the development of Artificial Intelligence (AI), some companies have started to explore the use of LLMs in practice (*e.g.*, IBM implementing AI processes in HR (BENJ EDWARDS, 2023)). Therefore, we propose a **C**ost-utility **A**nalysis **M**odel for Human-in-the-**LO**o**P** systems (CAMLOP ; Sec. 3) to analyze and balance the cost and utility. CAMLOP describes the cost-utility ratio that is introduced from the economics theory into the AI field to quantify these two factors (*i.e.*, cost and utility) and spread the two factors across various aspects (*e.g.*, time and money as cost; accuracy and user satisfaction as utility) so that reliable scores of various aspects are achieved.

We instantiate MCS with twelve datasets across three classes of tasks — arithmetic, commonsense, and symbolic reasoning (Sec. 4). MCS achieves new state-of-the-art levels of performance across most of the tasks. To show the applicability in real-world business, we apply CAMLOP to practice by posing an example to illustrate the balance between utility and cost in Sec. 4.5. Notably, a significant advantage w.r.t cost and utility proves our MCS 's superior over strong baselines.

2 Manual Correction System

MCS automatically finds the incorrectly predicted samples to indicate when humans should be involved (Sec. 2.1) and then provides efficient operations to indicate how to correct rationales (Sec. 2.2). Fig. 1 shows the whole four stages in MCS. The first and final stages are simple prompting. The intermediate filtering stage and correction stage are our focus, as detailed below.

2.1 Filtering Stage

As shown in Fig. 1, after the first stage, the LLM samples three plausible rationales for a math problem that arrive at different answers. Just like humans, LLMs may make countless and various mistakes, but there are only a limited number of correct rationales for the right result. If most of the sampled rationales cannot make agreements, with a high probability this sample is wrongly predicted. To empirically prove that, we conduct quantitative experiments and discover that incorrectly predicted samples tend to have greater diversity in their final answer when solving difficult reasoning problems. (Please refer to Appendix B for more details).

Specifically, the LLM is prompted with a set of manually written CoT exemplars following Wei et al. (2022) in the first stage. (Please refer to Appendix for more details) Then, we sample a set of candidate outputs from the LLM's decoder to generate a set of rationales[1]. Finally, we use the diversity degree to identify the most likely incorrect sample for humans to involve. Here, we adopt a widely-used method to select the samples: Diversity Entropy (Brinker, 2003; Yang et al., 2015; Agarwal et al., 2020). A further study about Diversity Entropy in Sec. 4.4 quantitatively demonstrates its advantage.

Formally, given a manually written CoT prompt and a sample $\mathbf{s}$, MCS decodes a set of N outputs, where each output $\mathbf{r}_i$ is a sequence of tokens representing the i-th rational, then the rational $\mathbf{r}_i$ is used to obtain the answer $\mathbf{a}_i$. As previously demonstrated, a greater diversity of the set of answers indicates potential incorrect predictions and flags a sample for humans to involve. First, we obtain the predicted answer $\mathbf{a}_i$ though $\arg\max_{\mathbf{a}_i} P(\mathbf{r}_i, \mathbf{a}_i \mid \mathbf{s})$. For example, in Fig. 1, $\mathbf{r}_i$ is *She has* $16 - 3 = 13$ *eggs left. So she has* $16 * 2 - 3 = \$13.$, and $\mathbf{a}_i$ is $\$13$. Then we calculate the answer distribution for the answer set $\{\mathbf{a}_{i,\cdots,N}\}$ of $\mathbf{s}$. For each distinct value $\mathbf{a} \in \{\mathbf{a}_{i,\cdots,N}\}$, the probability is as follows:

$$\mathbf{p_a} = \frac{\sum_{i=1}^{|N|} \mathbf{1}(\mathbf{a}_i = \mathbf{a})}{|N|} \tag{1}$$

where $|N|$ denotes the number of answers. For example, in Fig. 1, there are three answers as well as three rationales. We use the answer entropy as the Diversity Entropy (DE) score for the sample $\mathbf{s}$:

$$\mathbf{DE} = \sum_{\mathbf{a} \in \{\mathbf{a}_i\}} -\mathbf{p_a} \log \mathbf{p_a} \tag{2}$$

The higher the DE score, the more likely it needs manual correction. A threshold α is set for DE as the hyper-parameter.

2.2 Correction Stage

Referring to how humans should involve in the loop, the most straight-forward idea is humans handling the filtered samples while the LLM processes the rest samples. However, humans handling the sample as a whole problem is still labor-consuming, especially for those difficult mathematical problems. Due to this, we claim that humans should pay local attention to simple sub-logics in the rationale. Here, a sub-logic is typically a group of words that can stand alone as a complete thought in a complex rationale. We denote a sentence as a sub-logic.

To support our claim, there exist some premises. Firstly, an incorrect rationale could output the correct final answer after correcting the erroneous sub-logic in the rationale. To empirically prove that, we conduct quantitative experiments for twelve datasets and discover that in general up to 50% of errors of CoT indeed are caused by incorrect intermediate rationales. After correcting these 50% incorrect rationales, the final answers turn out to be correct. Secondly, correcting sub-logics indeed solves the majority of incorrect rationales. We conduct the analytical experiment across multiple tasks in Sec. 4.3 and provide the evidence. Thirdly, the questionnaire survey shows that correcting each sub-logic independently is much easier and more user-friendly for humans than checking the entire rationale (Please refer to Appendix C for more details).

Specifically, in the correction stage, we ask humans to check the filtered sample and only correct the rationale with the highest probability. During the correction, to simplify, the operations that a human can perform on the sub-logics include "modifying", "adding", and "deleting". As shown in Tab. 1, the first cause displays the modifying operation. After the modifying operation, the corrected sub-logic "$3 * 100 + 8 * 10 + 3 * 1 = 383$" helps the LLM output the correct answer.

[1] Most existing sampling algorithms including temperature sampling (Ackley et al., 1985; Ficler and Goldberg, 2017), top-k sampling (Fan et al., 2018; Holtzman et al., 2018; Radford et al., 2019) and nucleus sampling (Holtzman et al., 2019) could be used for sampling the required rationals. Here we follow Wang et al. (2022) for a fair comparison. Other sampling methods can also bring a general benefit.

Correction Operation: Modifying
QUESTION: Q: I have 3 hundred, 8 tens, and 3 ones. What number am I? A: RATIONALE: I have 3 hundred, 8 tens, and 3 ones. That means I have **«Before Modifying»:** $3*100+8*10+3*1=303$ **«After modifying»:** $3*100+8*10+3*1=383$.
Correction Operation: Deleting
QUESTION: Clarence has 5 oranges. He gets 3 more from Joyce. Later, Clarence buys 9 Skittles at the store. How many oranges does Clarence have in all? A: RATIONALE: Clarence has 5 oranges. He gets 3 more from Joyce, so now he has $5+3=8$ oranges. **«Delete»:** Later he buys 9 Skittles at the store, so he has $8-9=-1$ oranges.
Correction Operation: Adding
QUESTION: Q: There are 83 trees in a park. 36 of them are willows and the rest are oaks. How many more oaks than willows are there in the park? A: RATIONALE: There are 83 trees in the park. 36 of them are willows, and the rest are oaks. This means there are $83-36=47$ oaks in the park. There are 47 more oaks than willows. **«Add»:** There are 36 willows and 47 oaks in the park now, so there are $47-36=11$ more oaks than willows.

Table 1: Examples of manual correction for incorrect sub-logic. The operations that a human can perform on the rationales include modifying, adding, and deleting.

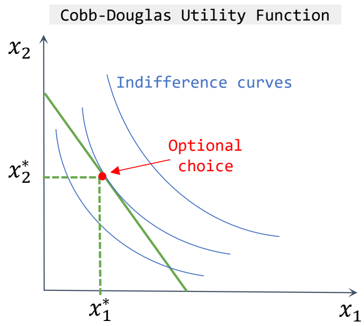

Figure 2: Illustration of CAMLOP.

3 Cost-utility Analysis Model for Human-in-the-Loop Systems

CAMLOP introduces the cost-utility relation that is introduced from the economics theory (Varian, 2014) into the AI field to quantify these two factors (*i.e.*, cost and utility). For human-in-the-loop systems like MCS , we divide the goods into two simple categories: human labor and LLM. Company strategic decision-makers always choose the best bundle of goods they can afford/cost. The costs include direct and indirect costs. The direct cost is the money the goods spent while indirect costs mainly include overhead costs from management and rent. Indirect costs also include intangible costs, such as the impact on customers, employees, or delivery times should be considered. Utilities include boosted accuracy, social prestige, and user satisfaction. For simplicity, we only consider money and time for cost while considering accuracy and user satisfaction for utility in our experiments.

We draw Fig. 2 where the horizontal axis x_1 and vertical axis x_2 are the quantity of human labor and LLMs respectively. First, we introduce notations related to the cost. We define $p_1 * x_1$ as the cost spent on human labor and $p_2 * x_2$ as the cost spent on the LLMs. We indicate the bundle by (x_1, x_2) (a data point in Fig. 2). The corresponding unit price is p_1 and p_2. The total cost the company decision-maker has to spend is denoted as y. Therefore, the budget constraint can be represented as $p_1x_1 + p_2x_2 \leqslant m$. The solid straight line is the set of data points that cost exactly y: $p_1x_1+p_2x_2 = m$. To note, the cost contains various aspects as mentioned before. In Fig. 2, for simplicity, we express these different aspects as a unified value according to a unified standard. Then we introduce utilities [2]. A utility function $u(x_1, x_2)$ is a way to assign a utility value to the bundle (x_1, x_2). As shown in Fig. 2, the set of all data points (x_1, x_2) such that $u(x_1, x_2)$ equals a constant is called a level set (solid curve). Those data points on higher indifference curves are getting larger utility. We adopted a commonly used utility function— Cobb-Douglas[3] utility function $u(x_1, x_2) = x_1^c x_2^d$, where c and d are positive numbers that we need to learn [4]. Given a model parameterized by c, d, and a fixed cost y, the model predicts the optimal choice (x_1^*, x_2^*) with the highest utility, which is desired by the company strategic decision-makers. Note an important feature of this optimal choice: at this data point the indifference curve is tangent to $p_1x_1 + p_2x_2 = y$.

To note, we introduce the modeling of CAMLOP in this section. More details about the inference and learning are shown in Appendix E and Appendix F.

4 Experiments

4.1 Setup

Tasks and datasets. For arithmetic reasoning tasks, we conducted a series of experiments on the Math Word Problem Repository (Amini et al., 2019), including AddSub (Hosseini et al., 2014), MultiArith (Roy and Roth, 2016), SingleEq

[2]Most notations are following those from (Varian, 2014)

[3]http://www.columbia.edu/~md3405/IM_recap_1_16.pdf

[4]Cobb-Douglas indifference curves is what economists referred as "well-behaved indifference curves". Cobb-Douglas utility functions are proved useful to present algebraic examples of economic field.

LLM	Method	AddSub	MultiArith	SingleEq	SingleOp	ASDiv	AQuA	SVAMP	GSM8K
GPT3-002	CoT-prompting	82.78	93.00	85.04	94.84	73.19	40.55	68.00	56.48
	Self-consistency	90.63	94.17	89.17	95.73	77.72	38.19	75.70	58.85
	MCS	92.15	**95.50**	92.51	96.62	75.52	**44.09**	74.60	61.56
	MCS + Self-con.	**97.22**	**95.50**	**94.09**	**98.75**	**79.63**	41.34	**80.10**	**62.92**
GPT3-003	CoT-prompting	88.86	94.00	94.49	94.31	79.58	47.24	79.10	59.51
	Self-consistency	91.65	97.83	96.26	95.55	84.11	50.39	83.10	63.99
	MCS	95.70	97.50	96.65	96.26	82.92	49.21	85.20	62.85
	MCS + Self-con.	**96.71**	**99.17**	**97.64**	**96.98**	**86.07**	**52.36**	**87.40**	**67.10**
ChatGPT	CoT-prompting	93.41	98.33	97.24	96.09	88.98	60.63	79.10	72.33
	Self-consistency	93.41	99.33	97.83	96.62	91.98	63.78	82.70	77.41
	MCS	95.95	99.50	97.83	96.62	90.94	61.02	83.00	74.53
	MCS + Self-con.	**96.20**	**99.83**	**98.23**	**96.98**	**93.37**	**64.17**	**85.50**	**79.08**

Table 2: Arithmetic reasoning accuracy by MCS and MCS + Self-consistency compared to Chain-of-Thought prompting and Self-consistency for LLM including GPT-3(text-davincci-002), GPT-3(text-davincci-003) and ChatGPT(gpt-3.5-turbo)

(Koncel-Kedziorski et al., 2015) and SingleOp (Kushman et al., 2014). We also included ASDiv (Miao et al., 2021), AQUA-RAT (Miao et al., 2021), GSM8K (Cobbe et al., 2021), and ASDiV (Patel et al., 2021). For commonsense reasoning tasks, we used CommonsensQA(Talmor et al., 2018) and StrategyQA(Geva et al., 2021). For symbolic reasoning tasks, we used Last Letter Concatenation and Coinflip(Wei et al., 2022)

Baselines. We primarily compare MCS with the following baselines. It is noteworthy that all baselines use the same LLM as the decoder. All of the annotators are undergraduate students who have basic math knowledge. For a fair comparison, we report the results of Self-consistency, MCS , and MCS + Self-consistency with the same 5 rationales sampled from the decoder. The details of the baselines are as follows:

1. *CoT-prompting.* Chain-of-thought prompting with greedy decoding (Wei et al., 2022).

2. *Self-consistency.* Chain-of-thought prompting replacing the greedy decoding strategy used in CoT-prompting. Self-consistency generates a set of rationales by sampling from LLM's decoder and determines the optimal answer by taking a majority vote (Wang et al., 2022).

Models and scales. We use GPT-3 (Ouyang et al., 2022; Brown et al., 2020b) and ChatGPT as the LLM. More details are provided in Appendix G. For our methods, we provide the following two variants:

1. *MCS* . MCS is the result of manual correction for the top 40% CoT predictions ranked out using DE. A detailed analysis of the threshold of Diversity Entropy is shown in Sec. 4.3.

2. *MCS +Self-consistency.* MCS + Self-consistency is the result of combining marginalizing out the sampled rationales with MCS . In practice, we use Self-consistency to get answers by majority vote, and then we use MCS to manually correct incorrect sub-logics of the first rationale out of decoded rationales with DE calculated based on the decoded rationales.

Sampling scheme. To sample diverse rationales, we followed similar settings to those used in Wang et al. (2022) for the open-text generation. We use $T = 0.7$ without top-k truncation. For a fair comparison, we use the same prompts as in Wei et al. (2022). The threshold of DE is set to be top 40%

4.2 Main Results

Arithmetic Reasoning The results are shown in Tab. 2. MCS generally improves the arithmetic reasoning performance at a large margin (4.68 points on average) compared with CoT. MCS + Self-consistency further improves the arithmetic reasoning performance (6.39 points on average). Especially for SingleEq and SVAMP, compared with CoT, the accuracy increased by 9.05 and 12.10 points, respectively.

Commonsense and Symbolic Reasoning Tab. 3 shows the results on commonsense and symbolic reasoning tasks. Similarly, MCS improves the performance and MCS + Self-consistency further boosts it. For symbolic reasoning, we adopt the out-of-distribution (OOD) setting where the in-

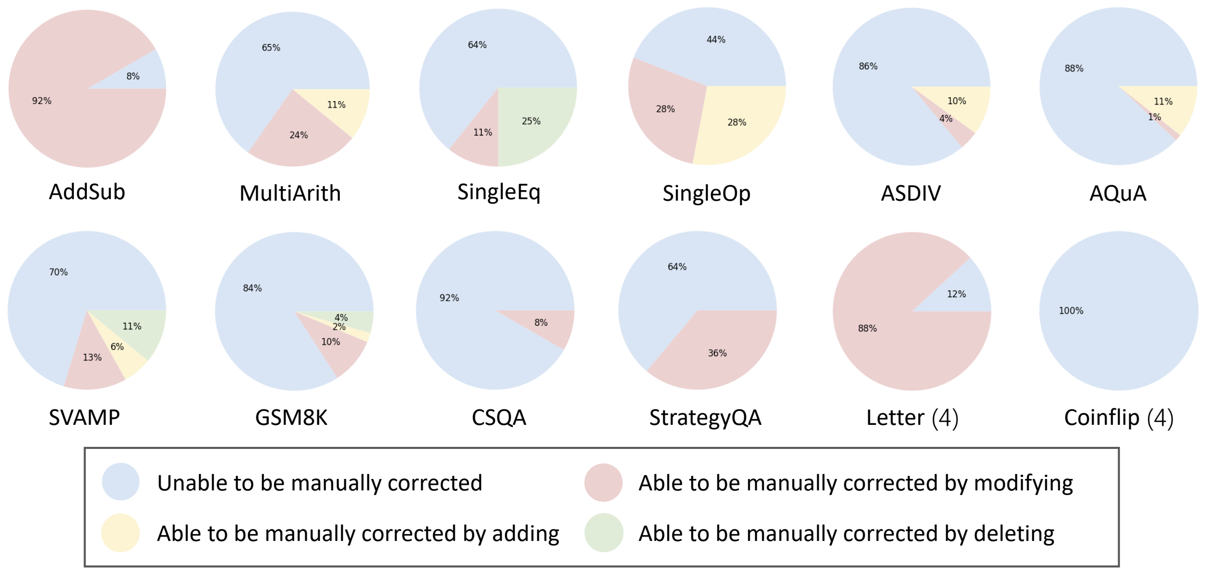

Figure 3: Illustration of error analysis of Chain of Thought Prompting across twelve tasks. Each error type is represented by a color. The share in color indicates the share of the error type.

put prompt contains samples of 4-letters and 4-flips (Wang et al., 2022) because this setting is more challenging. We do not adopt the in-distribution setting because GPT-3 can already achieve 100% accuracy with the in-distribution setting as shown in Wei et al. (2022). Even in difficult OOD setting, the gain of MCS +Self-consistency is significant compared to CoT-prompting and Self-consistency.

Model	*Commonsense*		*Symbolic*	
	CSQA	StraQA	Letter	Coinflip
CoT-prompting	72.32	60.13	49.20	81.40
Self-consistency	76.09	61.40	54.40	**93.20**
MCS	73.71	60.88	75.40	81.40
MCS + Self-con.	**77.07**	**62.23**	**78.40**	**93.20**

Table 3: Commonsense and symbolic reasoning accuracy. For each task, we report the median scores among 5 runs.

4.3 Analysis of Whether Correcting Sub-logics Solves the Majority of Incorrect Rationales

We conduct experiments on twelve datasets to check whether correcting sub-logics solves the majority of incorrect rationales. Each task is represented by a pie chart. For each task, we conduct the error analysis for CoT prompting and analyze the error types of rationales. We divided the error types into four categories: errors that are able to be corrected by the "modifying" operation, the "adding" operation, the "deleting" operation, and the rest of the errors that are unable to be manually corrected. The percentage of each type across datasets is shown in Fig. 3. More details are shown in Appendix C.2.

The first three categories constituent the majority of incorrect rationales and can be solved by correcting independent sub-logics instead of the whole rationale. More specifically, CoT often makes mistakes when calculating polynomial calculations with decimal points, which account for a large part of manual correction and can be corrected by the "modifying" operation. For the "adding" operation, it functions when CoT often fails to convert the units, for example, from grams to kilograms. CoT often outputs redundant logic, leading to incorrect answers, which could be fixed by the "deleting" operation. Except for the error mentioned above, errors that are unable to be manually corrected include misinterpretation of the question, incorrect formula, whole incorrect composition of sub-logics and so on.

Validation of Diversity Entropy Additionally, we find that the advantage of Self-consistency often comes from fixing the errors that are unable to be manually corrected. Sampling a large set of rationales and taking a majority vote helps the fix of misinterpretation of the question while making little help in fixing calculation error. On the contrary, MCS is beneficial for other three categories of errors including "modifying", "adding" and "deleting". The difference between Self-consistency and MCS illustrates why MCS + Self-consistency

Calculation Strategy	ASDiv	AQuA	SVAMP	GSM8K
Unnormalized Weighted Average	73.71	44.09	74.50	61.41
Normalized Weighted Average	73.71	40.94	**74.60**	**61.56**
Unnormalized Weighted Sum	73.80	42.52	74.50	60.20
Normalized Weighted Sum	73.37	**44.88**	71.30	59.21
Unnormalized Unweighted Sum (Majority Vote)	**75.52**	44.09	**74.60**	**61.56**

Table 4: Accuracy comparison of different strategies of computing answer probability. The threshold of Diversity Metrics is set to be top 40%.

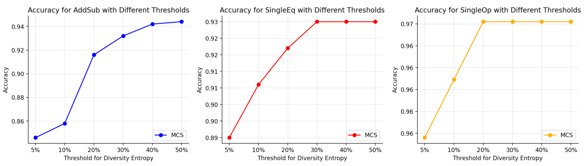

Figure 4: Results of different thresholds of DE. It shows the results of MCS with 5%, 10%, 20%, 30%, 40% and 50% DE for AddSub (Left), SingleEq (Medium) and SingleOp (Right). Results show that DE-based filtering is an efficient method to rank the possibility to be incorrect for the output of CoT predictions, and samples with incorrect output will be ranked higher than those without.

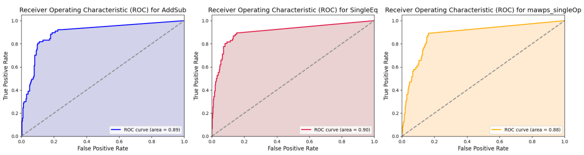

Figure 5: ROC Curves for DE to filter out the incorrect CoT outputs. It shows the ROC Curve for AddSub (Left), Singleeq (Medium) and SingleOp (Right). The results indicate that DE is a reliable metrics that can determine the samples most likely to be incorrectly predicted for humans to involve.

achieves great performance as shown in Tab. 2. Obviously, MCS and Self-consistency play different roles and be mutually complementary.

4.4 Additional Study

To validate the effectiveness of Diversity Entropy in determining whether the manual correction is necessary for each sample, we draw a ROC Curve in Fig. 5 to demonstrate its ability to rank the likelihood of incorrect outputs. The selection of the threshold involves a trade-off between performance and human labor. Fig. 4 shows that the performance stabilizes after reaching the threshold of top 20% to top 40% for most datasets. Therefore, we set the threshold to be top 40% across all our experiments. As the manual correction is labor-consuming and time-consuming, Diversity Entropy can help save time and labor by allowing humans to focus on checking only a small percentage.

Analysis of Aggregation Strategies The majority vote method of calculating the answer probability over all sampled rationales can be regarded as taking an unnormalized unweighted sum. As described in Wang et al. (2022), other methods of computing answer probability of $\mathbf{a}$ include the unnormalized weighted average, normalized weighted average, unnormalized weighted sum,

and normalized weighted sum. More details about the above calculation are provided in Appendix H. Tab. 4 shows that unnormalized unweighted sum generally outperforms others. We use this setting in experiments as Wang et al. (2022).

4.5 Balancing Cost and Utility

In this section, we conduct experiments on the SingleEq dataset to quantitatively calculate cost and utility for CAMLOP . For the cost, we consider money and time. We set the price of the LLM as $\mathbf{p}_{llm}$ and the time cost as $\mathbf{t}_{llm}$. Since we use GPT-3, the price $\mathbf{p}_{llm}$ for a single math problem (decoding once) is \$0.08 on average, and the time cost $\mathbf{t}_{llm}$ is 0.8 second based on empirical results [5]. The price of solving a single math problem with only human labor is $\mathbf{p}_{human}$ and the time cost is $\mathbf{t}_{human}$. We set $\mathbf{p}_{human}$ to be $\$0.125$ and $\mathbf{t}_{human}$ to be 60 seconds based on our empirical results. [6] The price of human labor for MCS to correct a single math problem $\mathbf{p}_{MCS}$ is $\$0.0625$ and the time cost $\mathbf{t}_{MCS}$ is

[5]The pricing of `text-davinci-002` is \$0.02 per 1000 tokens, which can be found at `https://openai.com/pricing`. We set $\mathbf{p}_{llm}$ to be \$0.08 because an input sample for few-shot CoT contains about 4000 tokens on average when decoding only once. Note that we only calculated the time for the main part (*i.e.*, the decoding) and ignored other parts that were fast enough to be ignored compared to the API calls.

[6]Minimum hourly wage in the United States is $\$7.5$, which can be found at `https://www.worker.gov/pay-for-hours-worked/`. Solving a problem requires 60 seconds on average. Therefore, the price and time cost required to complete a problem are $\$0.125$ and 60 seconds, respectively.

Plans	Time	Money	Acc.	Utility(User Satis.)
Human	60s	\$0.125	93.20	86.40
CoT Prompting	0.8s	\$0.080	85.04	81.60
Self-Consistency ($\mathbf{N}_{self}$ = 10)	8s	\$0.800	92.49	85.80
MCS ($\mathbf{N}_{MCS}$ = 5, $\alpha = 20\%$)	10.8s	\$0.4925	91.00	84.20
MCS + Self-consistency ($\mathbf{N}_{MCS}$ = 5, $\alpha = 20\%$)	10.8s	\$0.4925	93.50	88.80
MCS ($\mathbf{N}_{MCS}$ = 5, $\alpha = 40\%$)	16.8s	\$0.505	92.51	85.60
MCS + Self-consistency ($\mathbf{N}_{MCS}$ = 5, $\alpha = 40\%$)	16.8s	\$0.505	94.09	90.80

Table 5: Analysis of cost and utility for SingleEq. MCS + Self-consistency generally outperforms other methods with higher utility and acceptable cost. $\mathbf{N}$.: # sampled rationale. α: DE threshold. Acc.: Accuracy. User Satis.: User Satisfaction. More details are shown in Appendix I.

30 seconds based on empirical results. Note the time required to inspect and correct is less than the time needed to fully solve the entire problem, therefore $\mathbf{t}_{MCS} < \mathbf{t}_{human}$.

For the utility, we consider user satisfaction as the comprehensive score. We ask five users to write down their satisfaction levels and calculate the average [7]. We also perform regression analysis on user satisfaction based on LLM and Human and ultimately learn the utility function $\mathbf{u}(\mathbf{x}_{llm}, \mathbf{x}_{human}) = \mathbf{x}_{llm}^{2.05} * \mathbf{x}_{human}^{1.94}$. For more details, please refer to Appendix I.

We experiment on five candidate plans based on models from Sec. 4.2 and Sec. 4.4 (Fig. 4 and Fig. 6):

1. *Human*: A plan that requires only human labor, which costs $\mathbf{p}_{human}$ and $\mathbf{t}_{human}$ seconds.

2. *CoT-prompting*: A naive CoT plan that only requires GPT-3 for decoding only once, which costs $\mathbf{p}_{llm}$ and $\mathbf{t}_{llm}$ seconds.

3. *Self-consistency*: A Self-consistency plan that requires only LLMs to sample from the decoder $\mathbf{N}_{self}$ times, which will cost $\mathbf{N}_{self} * \mathbf{p}_{llm}$ and $\mathbf{N}_{self} * \mathbf{t}_{llm}$ seconds.

4. *MCS* : MCS samples from LLM decoder $\mathbf{N}_{MCS}$ times and uses top α as threshold, requiring $(\mathbf{N}_{MCS} + 1) * \mathbf{p}_{llm} + \alpha * \mathbf{p}_{MCS}$ and $(\mathbf{N}_{MCS} + 1) * \mathbf{t}_{llm} + \alpha * \mathbf{t}_{MCS}$ seconds.

5. *MCS + Self-consistency*: A MCS + Self-consistency plan that requires to sample from the decoder $\mathbf{N}_{MCS}$ times, which costs the same as the MCS plan.

[7] See Appendix for more details about user satisfaction. The impact of accuracy on user satisfaction is much larger than time cost, we speculate that most users care more about accuracy of solving problems than the time cost, as SingleEq is a math-solving dataset.

The results are shown in Tab. 5. The result shows that MCS +Self-consistency generally outperforms other methods with higher utility (*i.e.*, better user satisfaction) as well as an acceptable cost.

5 Related Work

5.1 Human-In-the-Loop System

Human-in-the-Loop system, aiming to achieve what neither humans nor machines can accomplish independently, is defined as a model requiring human interaction (Karwowski, 2006). When machines cannot solve the problem, or when cost or security considerations require humans to participate, manual intervention is necessary (Wu et al., 2022; Zanzotto, 2019; Mosqueira-Rey et al., 2023). Human-in-the-loop system outperforms both standalone AI and humans working alone (Bien et al., 2018).

5.2 Chain-of-Thought Prompting

Chain-of-Thought (CoT) prompting enables models to decompose multi-step problems into smaller steps. With CoT, LLMs can solve complex reasoning problems that cannot be solved with standard prompting methods (Wei et al., 2022; Wang et al., 2022). Despite its usefulness, CoT may be prone to errors, which can have a negative impact on the reasoning of the model.

6 Conclusion

We propose the MCS to explore how manual correction of rationales can improve LLM's reasoning ability. Then, we propose CAMLOP to quantitatively and systematically analyze and balance the cost and the corresponding utility. Experiments demonstrate that our MCS significantly outperforms strong baselines including the CoT prompting approach and Self-consistency approach.

7 Limitations

In this paper, we focused on the manual correction of the incorrect logic of the sampled output of Chain of Thought, without considering the mechanism of the fully automatic pipeline. As a machine-learning pipeline, human involvement may lead to additional human labor, which may be able to avoid by training a model to correct the incorrect reasoning paths.

References

David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. 1985. A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169.

Sharat Agarwal, Himanshu Arora, Saket Anand, and Chetan Arora. 2020. Contextual diversity for active learning. In *European Conference on Computer Vision*, pages 137–153. Springer.

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*.

BENJ EDWARDS. 2023. Ibm plans to replace 7,800 jobs with ai over time, pauses hiring certain positions, IBM CEO Arvind Krishna says he could see 30% of back-office functions replaced by AI over 5 years. https://arstechnica.com/information-technology/2023/05/ibm-pauses-hiring-around-7800-roles-that-could-be-replaced-by-ai/.

Nicholas Bien, Pranav Rajpurkar, Robyn L Ball, Jeremy Irvin, Allison Park, Erik Jones, Michael Bereket, Bhavik N Patel, Kristen W Yeom, Katie Shpanskaya, et al. 2018. Deep-learning-assisted diagnosis for knee magnetic resonance imaging: development and retrospective validation of mrnet. *PLoS medicine*, 15(11):e1002699.

Klaus Brinker. 2003. Incorporating diversity in active learning with support vector machines. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 59–66.

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. 2020a. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020b. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Bart Custers. 2022. Ai in criminal law: An overview of ai applications in substantive and procedural criminal law. *Law and Artificial Intelligence*, pages 205–223.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. *arXiv preprint arXiv:1707.02633*.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Google Research. 2023. Minerva: Solving quantitative reasoning problems with language models.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. 2018. Learning to write with cooperative discriminators. *arXiv preprint arXiv:1805.06087*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *EMNLP*, pages 523–533. Citeseer.

Waldemar Karwowski. 2006. *International Encyclopedia of Ergonomics and Human Factors, -3 Volume Set*. Crc Press.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 271–281.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2021. A diverse corpus for evaluating and developing english math word problem solvers. *arXiv preprint arXiv:2106.15772*.

Eduardo Mosqueira-Rey, Elena Hernández-Pereira, David Alonso-Ríos, José Bobes-Bascarán, and Ángel Fernández-Leal. 2023. Human-in-the-loop machine learning: A state of the art. *Artificial Intelligence Review*, 56(4):3005–3054.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Subhro Roy and Dan Roth. 2016. Solving general arithmetic word problems. *arXiv preprint arXiv:1608.01413*.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

Nan Shao, Zefan Cai, Chonghua Liao, Yanan Zheng, Zhilin Yang, et al. 2023. Compositional task representations for large language models. In *The Eleventh International Conference on Learning Representations*.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*.

Hal R. Varian. 2014. *Intermediate microeconomics: a modern approach*. New York :W.W. Norton Company.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Xingjiao Wu, Luwei Xiao, Yixuan Sun, Junhang Zhang, Tianlong Ma, and Liang He. 2022. A survey of human-in-the-loop for machine learning. *Future Generation Computer Systems*.

Yi Yang, Zhigang Ma, Feiping Nie, Xiaojun Chang, and Alexander G Hauptmann. 2015. Multi-class active learning by uncertainty sampling with diversity maximization. *International Journal of Computer Vision*, 113(2):113–127.

Fabio Massimo Zanzotto. 2019. Human-in-the-loop artificial intelligence. *Journal of Artificial Intelligence Research*, 64:243–252.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

A Related Work

A.1 Human-In-the-Loop System

Human-in-the-Loop system, aiming to achieve what neither humans nor machines can accomplish independently, is defined as a model requiring human interaction (Karwowski, 2006). When machines cannot solve the problem, or when cost or security considerations require humans to participate, manual intervention is necessary (Wu et al., 2022; Zanzotto, 2019; Mosqueira-Rey et al., 2023). Human-in-the-loop system outperforms both standalone AI and humans working alone (Bien et al., 2018).

Recently, LLM-based AI (Artificial Intelligence) systems are developing very quickly, and this trend is expected to expand to the majority of the workforce in the near future (Ouyang et al., 2022; Zhang et al., 2022; Sanh et al., 2021). However, these systems do not always provide satisfactory answers without human intervention, especially mathematical problems. Additionally, in domains such as criminal fact identification and charge predictions, inference should be reasonable and controlled by humans (Custers, 2022) while LLMs are not qualified. Different from ChatGPT's RLHF (Reinforcement Learning from Human Feedback), we take the first step to use human feedback in an online way without access to parameters. Even though it's a preliminary step, this online method could benefit from further refinement and combination with RLHF in future research.

A.2 Chain-of-Thought Prompting

Chain-of-Thought (CoT) prompting enables models to decompose multi-step problems into smaller steps. With CoT, LLMs can solve complex reasoning problems that cannot be solved with standard prompting methods (Wei et al., 2022; Wang et al., 2022). Despite its usefulness, CoT may be prone to errors, which can have a negative impact on the reasoning of the model. Fortunately, most mistakes can be easily interpreted. About half of these mistakes are related to incorrect calculations while the other half are mistakes from flawed reasoning where rationales lack the necessary knowledge (Google Research, 2023). To address this issue, we limit users to modifying, deleting, or adding a single sub-logic as a means of resolving both types of errors. Additionally, we have found that most mistakes can be easily detected and corrected by humans through rationales. Against this background, CoT presents an opportunity for humans to modify predicted outcomes through sub-logics of rationales.

B Experiments for Filtering Stage

After the first stage, the LLM samples plausible rationales for a problem that arrive at different answers. Just like humans, LLMs may make countless and various mistakes, but there are only a limited number of correct rationales for the right result. If most of the sampled rationales cannot make agreements, with a high probability this sample is wrongly predicted. To empirically prove that, we conduct quantitative experiments and discover that incorrectly predicted samples tend to have greater diversity in their final answer when solving difficult reasoning problems.

Specifically, the LLM is prompted with a set of manually written CoT exemplars following Wei et al. (2022) in the first stage. Then, we sample a set of 5 candidate outputs from the LLM's decoder to generate a set of rationales. Based on the sampled rationales, we divide the samples into two parts: **Part** 1 has all sampled rationales pointing to the same final answer (*i.e.*, the Diversity Entropy score as Sec. 2.1 of such samples should be equal to 0); **Part 2** has sampled rationales pointing to different final answers, which is the part outside the first part of samples (*i.e.*, the Diversity Entropy score as Sec. 2.1 of such samples should be greater than 0). Next, we calculate the accuracy of **Part 1** and **Part 2** for each dataset separately. We use the first answer of each sample as the result of CoT-Prompting and use all five answers to calculate the Diversity Entropy score. The results are shown in Tab. 6, Tab. 7, Tab. 8 and Tab. 9. The accuracy of **Part 1** is generally larger than **Part 2**. It demonstrates the superiority of Diversity Entropy and experimentally confirms the intuition that incorrectly predicted samples tend to have greater diversity in their final answer when solving difficult reasoning problems.

C Experiments for Correction Stage

C.1 Incorrect Rationale Could Output the Correct Final Answer after Manually Correcting the Erroneous Rationale.

An incorrect rationale could output the correct final answer after correcting the erroneous rationale. To empirically prove this, we conduct quantitative experiments for twelve datasets and discover that in general most of the errors of CoT indeed are caused by incorrect rationales. After correcting these incorrect rationales, the final answers turn out to be correct.

Specifically, we explored the limits of the CoT-based methods (namely CoT-Prompting, Self-Consistency, and MCS) when humans correct rationales while disregarding cost. Humans were instructed to thoroughly check all samples and ensure the correctness of all rationales. Tables 10 and 11 present the results, where the upper bound of CoT-Prompting is denoted as CoT-Upperbound and the upper bound of Self-Consistency is denoted as SC-Upperbound. Self Consistency and MCS+Self Consistency have the same upper bound in extreme cases (*i.e.*, the threshold of Diversity Entropy score is set to 100%) while CoT-Upperbound and MCS have the same upper bound in extreme cases (*i.e.*, the threshold of Diversity Entropy score is set to 100%). The experimental results demonstrate that the upper bounds are quite high, indicating that an incorrect rationale could produce the correct final answer after correcting the errors. To note, this

Method	Part	*Arithmetic Reasoning*								
		AddSub			MultiArith			SingleEq		
		Num.	Ratio	Acc.	Num.	Ratio	Acc.	Num.	Ratio	Acc.
CoT-Prompting	Part 1	245	62.03%	97.55	299	49.83%	100.00	369	72.64%	97.83
	Part 2	150	37.97%	53.33	301	50.17%	82.39	139	27.36%	51.08
	Part 1&2	395	100.00%	82.78	600	100.00%	93.00	508	100.00%	85.04
Self-Consistency	Part 1	245	62.03%	97.55	299	49.83%	100.00	369	72.64%	97.83
	Part 2	150	37.97%	71.33	301	50.17%	87.38	139	27.36%	66.19
	Part 1&2	395	100.00%	90.63	600	100.00%	94.17	508	100.00%	89.17

Table 6: Analysis for Diversity Entropy in Filtering Stage (I). The accuracy of **Part 1** is generally larger than **Part 2**. The result demonstrates the superiority of Diversity Entropy and experimentally confirms the intuition that incorrectly predicted samples tend to have greater diversity in their final answer when solving difficult reasoning problems. For each task, we report the median scores among 5 runs.

Method	Part	*Arithmetic Reasoning*								
		SingleOp			ASDiv			AQuA		
		Num.	Ratio	Acc.	Num.	Ratio	Acc.	Num.	Ratio	Acc.
CoT-Prompting	Part 1	423	75.27%	98.35	1122	53.53%	96.88	48	18.90%	52.08
	Part 2	139	24.73%	58.99	974	46.47%	42.51	206	81.10%	37.38
	Part 1&2	562	100.00%	94.84	2096	100.00%	73.19	254	100.00%	40.55
Self-Consistency	Part 1	423	75.27%	98.35	1122	53.53%	96.88	48	18.90%	52.08
	Part 2	139	24.73%	70.50	974	46.47%	52.78	206	81.10%	32.04
	Part 1&2	562	100.00%	95.73	2096	100.00%	77.72	254	100.00%	38.19

Table 7: Analysis for Diversity Entropy in Filtering Stage (II). The accuracy of **Part 1** is generally larger than **Part 2**. The result demonstrates the superiority of Diversity Entropy and experimentally confirms the intuition that incorrectly predicted samples tend to have greater diversity in their final answer when solving difficult reasoning problems. For each task, we report the median scores among 5 runs.

limitation represents only the upper bounds of our method, and its practical implementation would require significant time and resources.

C.2 Correcting Erroneous Sub-logic Indeed Solves the Majority of Erroneous Rationale.

Correcting erroneous sub-logic indeed solves the majority of erroneous rationale. We conduct the analytical experiment across multiple tasks in Sec. 4.3 and provide the evidence.

We conduct experiments on twelve datasets to check whether correcting sub-logics solves the majority of incorrect rationales. Each task is represented by a pie chart. For each task, we conduct the error analysis for CoT prompting and analyze the error types of rationales. We divided the error types into four categories: errors that are able to be corrected by the "modifying" operation, the "adding" operation, the "deleting" operation, and the rest of the errors that are unable to be manually corrected. The percentage of each type across datasets is shown in Fig. 3.

Sec. 4.3 presents experiments in Fig. 3 on twelve datasets to check whether correcting sub-logics solves the majority of erroneous rationales. Figure 3 illustrates the error analysis of the CoT Prompting across twelve tasks. We list the detailed numbers of the error analysis in Tab. 12 and Tab. 13. Results show that correcting erroneous sub-logic indeed solves the majority of erroneous rationale (*i.e.*, each erroneous rationale indeed can be corrected by only editing a single erroneous sub-logic).

C.3 Correcting Each Sub-logics Independently is Much Easier and More User-friendly than Correcting the Entire Rationale

We conduct the human evaluation. The questionnaire survey shows that correcting each sub-logic independently (*i.e.*, our approach) is much easier and more user-friendly than checking the entire rationale. We present the time that humans need to check and correct the incorrect sub-logics compared to correcting the entire rationale as Tab. 14 and Tab. 15.

The result presents the average time (seconds) needed for a human to check and correct the incor-

Method	Part	*Arithmetic Reasoning*						*Commonsense Reasoning*		
		SVAMP			GSM8K			CSQA		
		Num.	Ratio	Acc.	Num.	Ratio	Acc.	Num.	Ratio	Acc.
CoT-Prompting	Part 1	438	43.80%	92.92	256	19.41%	93.36	792	64.86%	85.98
	Part 2	562	56.20%	47.86	1063	80.59%	47.70	429	35.14%	47.09
	Part 1&2	1000	100.00%	68.00	1319	100.00%	56.48	1221	100.00%	72.32
Self-Consistency	Part 1	438	43.80%	92.92	256	19.41%	93.36	792	64.86%	85.98
	Part 2	562	56.20%	62.46	1063	80.59%	50.71	429	35.14%	57.81
	Part 1&2	1000	100.00%	75.70	1319	100.00%	58.85	1221	100.00%	76.09

Table 8: Analysis for Diversity Entropy in Filtering Stage (III). The accuracy of **Part 1** is generally larger than **Part 2**. The result demonstrates the superiority of Diversity Entropy and experimentally confirms the intuition that incorrectly predicted samples tend to have greater diversity in their final answer when solving difficult reasoning problems. For each task, we report the median scores among 5 runs.

Method	Part	*Commonsense Reasoning*			*Symbolic Reasoning*					
		StrategyQA			Letter (4)			Coinflip (4)		
		Num.	Ratio	Acc.	Num.	Ratio	Acc.	Num.	Ratio	Acc.
CoT-Prompting	Part 1	1502	65.88%	66.31	175	35.00%	72.00	384	38.40%	98.70
	Part 2	778	34.12%	48.59	325	65.00%	36.31	616	61.60%	69.48
	Part 1&2	2280	100.00%	60.13	500	100.00%	49.20	1000	100.00%	81.40
Self-Consistency	Part 1	1502	65.88%	66.31	175	35.00%	72.00	384	38.40%	98.70
	Part 2	778	34.12%	52.57	325	65.00%	44.62	616	61.60%	89.61
	Part 1&2	2280	100.00%	61.40	500	100.00%	54.40	1000	100.00%	93.20

Table 9: Analysis for Diversity Entropy in Filtering Stage (IV). The accuracy of **Part 1** is generally larger than **Part 2**. The result demonstrates the superiority of Diversity Entropy and experimentally confirms the intuition that incorrectly predicted samples tend to have greater diversity in their final answer when solving difficult reasoning problems. For each task, we report the median scores among 5 runs.

rect sub-logics compared to correcting the entire rationale for each sample. The time humans need to check and correct the incorrect sub-logics is much less than the time needed to correct the entire rationale for each sample, proving that correcting each sub-logic independently is much easier and more user-friendly for humans than checking the entire rationale.

D Analysis of the Number of Sampled Rationales

We test the accuracy with respect to varying the number of rationales (*i.e.*, 5, 10, 15, 20, 25, 30, 35, 40) in Fig. 6. The results are arithmetic reasoning accuracy on SingleEq. For a fair comparison, both MCS and Self-consistency use the same prompts as in Wei et al. (2022). Both MCS and Self-consistency use the same 5 rationales sampled from the decoder. In our experiments, the threshold of Diversity Metrics is set to be top 40%. The results show that MCS generally outperforms self-consistency and benefits from the increasing number of sampled rationales.

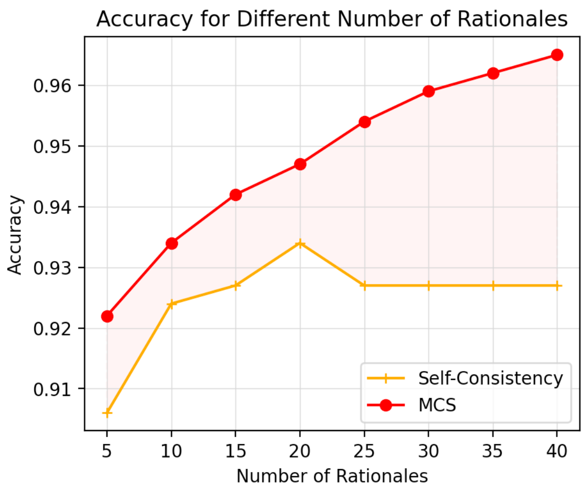

Figure 6: Experiments of different numbers of rationales.

Model	*Arithmetic Reasoning*							
	AddSub	MultiArith	SingleEq	SingleOp	ASDiv	AQuA	SVAMP	GSM8K
CoT-Prompting	82.78	93.00	85.04	94.84	73.19	40.55	68.00	56.48
CoT-Upperbound	97.72	96.33	94.09	96.80	75.62	47.64	77.50	63.76
Self-Consistency	90.63	94.17	89.17	95.73	77.72	38.19	75.70	58.85
SC-Upperbound	98.48	96.33	95.67	98.93	81.58	44.49	82.00	64.67

Table 10: Upperbound Analysis of CoT-Prompting, Self-Consistency and MCS (I). The experimental results demonstrate that the upper bounds are quite high, indicating that an incorrect rationale could produce the correct final answer after correcting the errors. To note, this limitation represents only the upper bounds of our method, and its practical implementation would require significant time and resources. For each task, we report the median scores among 5 runs.

Model	*Commonsense*		*Symbolic*	
	CSQA	StraQA	Letter	Coinflip
CoT-Prompting	72.32	60.13	49.20	81.40
CoT-Upperbound	74.61	60.88	93.80	81.40
Self-Consistency	76.09	61.40	54.40	93.20
SC-Upperbound	77.97	62.23	96.00	93.20

Table 11: Upperbound Analysis of CoT-Prompting, Self-Consistency and MCS (II). The experimental results demonstrate that the upper bounds are quite high, indicating that an incorrect rationale could produce the correct final answer after correcting the errors. To note, this limitation represents only the upper bounds of our method, and its practical implementation would require significant time and resources. For each task, we report the median scores among 5 runs.

E Inference for CAMLOP

Given a model parameterized by c, d, and a fixed cost y, the model predicts the optimal choice (x_1^*, x_2^*) with the highest utility, which is desired by the company strategic decision-makers. Note an important feature of this optimal choice: at this data point (namely, optimal choice point) the indifference curve is tangent to $p_1 x_1 + p_2 x_2 = y$. According to this feature, the inference is to get (x_1^*, x_2^*) that satisfied the following equation:

$$u'(x_1^*, x_2^*) = -\frac{p_1}{p_2} \qquad (3)$$

which will derive the optimal choice (x_1^*, x_2^*):

$$x_1^* = \frac{c}{c+d}\frac{m}{p_1}, x_2^* = \frac{d}{c+d}\frac{m}{p_2} \qquad (4)$$

F Learning for CAMLOP

We have seen how to make the best decision based on the inference of CAMLOP. But in real life we have to work the other way around: we observe some historical cost and utility datapoints, but our problem is to estimate what kind of utility function is induced from the observations.

Concretely, suppose that we observe a number of industries making choices between LLMs and human workers based on their considerations of commute times, money costs, accuracy, *etc*. There exists an analytic solution of c, d obtained by statistical techniques that best fit the observed data points. In this way, the historical datapoints give a way to estimate the utility function. More specifically, we use regression analysis to find the utility function that best describes the relation between x and utility. Mean square error is typically employed as the loss function for learning the utility function. The loss function is defined on J training datapoints $X = \{(x_1^{(1)}, x_2^{(1)}), (x_1^{(2)}, x_2^{(2)}), ..., (x_1^{(J)}, x_2^{(J)})\}$:

$$L(c, d) = \frac{1}{J}\sum_{i=1}^{J} \log u(x_1^{(i)}, x_2^{(i)}; c, d) \qquad (5)$$

where the model parameters are c, d. A normal equation or gradient descent can be used to optimize this loss function and obtain the final c, d.

G Experiment Details

We choose GPT-3 because of its superior CoT reasoning performance, as reported in the work of Wei et al. (2022) and Wang et al. (2022). Due to the limited context window size (up to 4096 word-pieces for the GPT-3 series of models), we use an 8-shot setting for all datasets. Our experiments are based on access to the OpenAI GPT-3 API. We perform all experiments in the few-shot setting, without training or fine-tuning the LLM. For a fair comparison, we use the same prompts as in the work of Wei et al. (2022). For arithmetic reasoning tasks, we use the same set of 8 manually written exemplars. For commonsense reasoning tasks, exemplars are randomly selected from the training set with manually written CoT prompts.

Operation	*Arithmetic Reasoning*											
	AddSub		MultiArith		SingleEq		SingleOp		ASDiv		AQuA	
	Num.	Ratio	Num.	Ratio	Num.	Ratio	Num.	Ratio	Num.	Ratio	Num.	Ratio
Modifying	33	92%	22	24%	3	11%	19	28%	15	4%	2	1%
Adding	0	0%	10	11%	0	0%	19	28%	38	10%	16	16%
Deleting	0	0%	0	0%	7	25%	0	0%	0	0%	0	0%
Unable	3	8%	60	65%	18	64%	30	44%	327	86%	132	88%

Table 12: Detailed numbers of the error analysis (I). The results are the detailed numbers of Fig. 3.

Operation	*Arithmetic Reasoning*				*Commonsense Reasoning*				*Symbolic Reasoning*			
	SVAMP		GSM8K		CSQA		StraQA		Letter (4)		Conflip (4)	
	Num.	Ratio	Num.	Ratio	Num.	Ratio	Num.	Ratio	Num.	Ratio	Num.	Ratio
Modifying	41	13%	54	10%	28	8%	39	36%	223	88%	0	0%
Adding	19	6%	11	2%	0	0%	0	0%	0	0%	0	0%
Deleting	35	11%	25	4%	0	0%	0	0%	0	0%	0	0%
Unable	225	70%	478	84%	310	92%	69	64%	30	12%	186	100%

Table 13: Detailed numbers of the error analysis (II). The results are the detailed numbers of Fig. 3.

We list the exact set of prompts used for all arithmetic reasoning tasks in Tab. 16, since there are multiple sets of prompts introduced in Wei et al. (2022). The prompts for CommonsenseQA and StrategyQA are the same as used in Wei et al. (2022).

H Diversity Metrics Over Diverse Reasoning Paths

As described in Sec. 4.4, the majority vote method of calculating the answer probability over all sampled rationales can be regarded as taking an unnormalized unweighted sum. As described in Wang et al. (2022), other methods of computing answer probability of $\mathbf{a}$ include the unnormalized weighted average, normalized weighted average, unnormalized weighted sum, and normalized weighted sum. Tab. 4 shows that unnormalized unweighted sum generally outperforms others. We use this setting in all experiments following Wang et al. (2022).

In practice, the majority vote method of calculating the answer probability over all sampled rationales proposed at Eq. 1 is the same as taking the unweighted sum over $\mathbf{a}_i$ (*i.e.*, $\sum_{i=1}^{|N|} \mathbf{1}(\mathbf{a}_i = \mathbf{a})$), where $|N|$ denotes the number of answers (*i.e.*, the number of sampling times). As described in Wang et al. (2022), another selection of computing answer probability of $\mathbf{a}$ over all sampled rationales is to use unnormalized probability $\mathbf{p}_{\mathbf{a}_i}$ of the language model generating $\mathbf{a}_i$ given the prompt of sample $\mathbf{s}$:

$$\mathbf{p}_{\mathbf{a}_i} = P(\mathbf{r}_i, \mathbf{a}_i \mid \mathbf{s}) \tag{6}$$

Then we use all unnormalized probability $\mathbf{p}_{\mathbf{a}_i}$ given by the language model's decoder to calculate the probability $\mathbf{p}_{\mathbf{a}}$ of the answer $\mathbf{a}$ for sample $\mathbf{s}$:

$$\mathbf{p}_{\mathbf{a}} = \frac{\sum_{i=1}^{|N|} \mathbf{1}(\mathbf{a}_i = \mathbf{a}) \mathbf{p}_{\mathbf{a}_i}}{|N|} \tag{7}$$

where $|N|$ denotes the number of rationales decoded for the sample $\mathbf{s}$. The result of using the calculation output of Eq. 7 as the probability of answer $\mathbf{a}$ is shown in Tab. 4 as **Unnormalized Weighted Sum** . Apart from computing $\mathbf{p}_{\mathbf{a}}$ by taking the unnormalized probability of the language

Human Operation	*Arithmetic Reasoning*							
	AddSub	MultiArith	SingleEq	SingleOp	ASDiv	AQuA	SVAMP	GSM8K
Correcting sub-logics	21s	24s	30s	14s	26s	62s	16s	45s
Correcting entire rationale	49s	80s	60s	32s	44s	102s	48s	77s

Table 14: Time (seconds) spent for correcting the incorrect sub-logics compared to correcting the entire rationale (I). The time humans need to check and correct the incorrect sub-logics is much less than the time needed to correct the entire rationale for each sample, proving that correcting each sub-logic independently is much easier and more user-friendly for humans than checking the entire rationale.

Human Operation	*Commonsense*		*Symbolic*	
	CSQA	StraQA	Letter	Coinflip
Correcting sub-logics	14s	24s	18s	36s
Correcting entire rationale	26s	36s	28s	40s

Table 15: Time (seconds) spent for correcting the incorrect sub-logics compared to correcting the entire rationale (II). The time humans need to check and correct the incorrect sub-logics is much less than the time needed to correct the entire rationale for each sample, proving that correcting each sub-logic independently is much easier and more user-friendly for humans than checking the entire rationale.

Q: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?
A: We start with 15 trees. Later we have 21 trees. The difference must be the number of trees they planted. So, they must have planted 21 - 15 = 6 trees. The answer is 6.
Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?
A: There are 3 cars in the parking lot already. 2 more arrive. Now there are 3 + 2 = 5 cars. The answer is 5.
Q: Leah had 32 chocolates and her sister had 42. If they ate 35, how many pieces do they have left in total?
A: Leah had 32 chocolates and Leah's sister had 42. That means there were originally 32 + 42 = 74 chocolates. 35 have been eaten. So in total they still have 74 - 35 = 39 chocolates. The answer is 39.
Q: Jason had 20 lollipops. He gave Denny some lollipops. Now Jason has 12 lollipops. How many lollipops did Jason give to Denny?
A: Jason had 20 lollipops. Since he only has 12 now, he must have given the rest to Denny. The number of lollipops he has given to Denny must have been 20 - 12 = 8 lollipops. The answer is 8.
Q: Shawn has five toys. For Christmas, he got two toys each from his mom and dad. How many toys does he have now?
A: He has 5 toys. He got 2 from mom, so after that he has 5 + 2 = 7 toys. Then he got 2 more from dad, so in total he has 7 + 2 = 9 toys. The answer is 9.
Q: There were nine computers in the server room. Five more computers were installed each day, from monday to thursday. How many computers are now in the server room?
A: There are 4 days from monday to thursday. 5 computers were added each day. That means in total 4 * 5 = 20 computers were added. There were 9 computers in the beginning, so now there are 9 + 20 = 29 computers. The answer is 29.
Q: Michael had 58 golf balls. On tuesday, he lost 23 golf balls. On wednesday, he lost 2 more. How many golf balls did he have at the end of wednesday?
A: Michael initially had 58 balls. He lost 23 on Tuesday, so after that he has 58 - 23 = 35 balls. On Wednesday he lost 2 more so now he has 35 - 2 = 33 balls. The answer is 33.
Q: Olivia has $23. She bought five bagels for $3 each. How much money does she have left?
A: She bought 5 bagels for $3 each. This means she spent 5 * $3 = $15 on the bagels. She had $23 in beginning, so now she has $23 - $15 = $8. The answer is 8.

Table 16: Few-shot exemplars for arithmetic reasoning tasks.

model generating $(\mathbf{r}_i, \mathbf{a}_i)$ given $\mathbf{s}$, we can normalize the output probability for $(\mathbf{r}_i, \mathbf{a}_i)$ by the output length of $\mathbf{r}_i$ (Brown et al., 2020b):

$$\mathbf{p}_{\mathbf{a}_i} = \exp^{\frac{1}{K}\sum_{k=1}^{K} \log p_{t_k}} \tag{8}$$

where p_{t_k} is the log probability of generating the k-th token t_k in $(\mathbf{r}_i, \mathbf{a}_i)$ conditioned on the previous tokens, and K is the total number of tokens in $(\mathbf{r}_i, \mathbf{a}_i)$:

$$p_{t_k} = P(t_k \mid \mathbf{s}, t_1, \ldots, t_{k-1}) \tag{9}$$

The result of using the calculation output of Eq. 8 as the normalized probability $\mathbf{p}_i^{\mathbf{a}}$ of the language model generating $\mathbf{a}_i$ given prompt of sample $\mathbf{s}$ is shown in Tab. 4 as **Normalized Weighted Sum**.

In addition, in Tab. 4 we also report the results by taking a weighted average, which means calculating a score for each $\mathbf{a}$ of its weighted sum divided by $\sum_{i=1}^{|N|} \mathbf{1}(\mathbf{a}_i = \mathbf{a})$.

Tab. 4 shows that unnormalized unweighted sum generally outperforms others. We use this setting in all experiments following Wang et al. (2022).

I Details of Balancing Cost and Utility

In Sec 5, we conduct experiments on the SingleEq dataset to quantitatively calculate cost and utility for CAMLOP . The trends on other datasets are consistent with SingleEq dataset. We randomly selected one dataset as an example to demonstrate the superiority of MCS in balancing cost and utility.

For the cost, we consider money and time. We set the price of the LLM as $\mathbf{p}_{llm}$ and the time cost as $\mathbf{t}_{llm}$. Since we use GPT-3, the price $\mathbf{p}_{llm}$ for a single math problem (decoding once) is \$0.08 on average, and the time cost $\mathbf{t}_{llm}$ is 0.8 second based on empirical results [8]. The price of solving a single math problem with only human labor is $\mathbf{p}_{human}$ and the time cost is $\mathbf{t}_{human}$. We set $\mathbf{p}_{human}$ to be \$0.125 and $\mathbf{t}_{human}$ to be 60 seconds based on our empirical results. [9] The price of human labor for MCS to correct a single math problem $\mathbf{p}_{MCS}$ is \$0.0625 and the time cost $\mathbf{t}_{MCS}$ is 30 seconds based on empirical results. Note the time required to inspect and correct is less than the time needed to fully solve the entire problem, therefore $\mathbf{t}_{MCS} < \mathbf{t}_{human}$.

For the utility, we consider user satisfaction as the comprehensive score. We ask five users to write down their satisfaction levels and calculate the average. The human ratings are collected via Amazon Turk. In addition to the effective data collected from 5 users for each evaluation method, data from several users were excluded due to failures in the attention verification. The hourly salary is \$10 per hour and per user. We randomly select a set of examples and the satisfaction level is rated from 1 to 5, with 1 as the worst satisfaction and 5 as the most user-friendly and best satisfaction. The human rating scores are then averaged.

We experiment on candidate plans based on models from Sec. 4.2 and Sec. 4.4 (Fig. 4 and Fig. 6), and the results are shown in Tab. 17. The calculation of time and money in Tab. 17 is shown as below:

1. *Human*: A plan that requires only human labor, which costs $\mathbf{p}_{human}$ and $\mathbf{t}_{human}$ seconds. So the time needed is $\mathbf{t}_{human} = 60$seconds, and the money needed is $\mathbf{p}_{human} = \$0.125$

2. *CoT-prompting*: A naive CoT plan that only requires GPT-3 for decoding only once, which costs $\mathbf{p}_{llm}$ and $\mathbf{t}_{llm}$ seconds. So the money needed is $\mathbf{p}_{llm} = \$0.08$ and the time needed is $\mathbf{t}_{llm} = 0.8$second.

3. *Self-consistency ($\mathbf{N}_{self} = 10$)*: A Self-consistency plan that requires only LLMs to sample from the decoder $\mathbf{N}_{self}$ times, which will cost $\mathbf{N}_{self} * \mathbf{p}_{llm}$ and $\mathbf{N}_{self} * \mathbf{t}_{llm}$ seconds. For $\mathbf{N}_{self} = 10$, the money needed is $\mathbf{N}_{self} * \mathbf{p}_{llm} = 10 * \$0.08 = \$0.8$, the time needed is $\mathbf{N}_{self} * \mathbf{t}_{llm} = 10 * 0.8 = 8$seconds.

4. *MCS ($\mathbf{N}_{MCS} = 5$, $\alpha = 20\%$)*: MCS samples from LLM decoder $\mathbf{N}_{MCS}$ times and uses top α as threshold, requiring $(\mathbf{N}_{MCS} + 1) * \mathbf{p}_{llm} + \alpha * \mathbf{p}_{MCS}$ and $(\mathbf{N}_{MCS} + 1) * \mathbf{t}_{llm} + \alpha * \mathbf{t}_{MCS}$ seconds. For $\mathbf{N}_{MCS} = 5$, $\alpha = 20\%$, the money needed is $(\mathbf{N}_{MCS} + 1) * \mathbf{p}_{llm} + \alpha * \mathbf{p}_{MCS} = \$0.08 * 6 + 20\% * \$0.0625 = \$0.4925$, the time needed is $(\mathbf{N}_{MCS} + 1) * \mathbf{t}_{llm} + \alpha * \mathbf{t}_{MCS} = 0.8 * 6s + 20\% * 30s = 10.8$ seconds.

[8] The pricing of `text-davinci-002` is \$0.02 per 1000 tokens, which can be found at `https://openai.com/pricing`. We set $\mathbf{p}_{llm}$ to be \$0.08 because an input sample for few-shot CoT contains about 4000 tokens on average when decoding only once. Note that we only calculated the time for the main part (*i.e.*, the decoding) and ignored other parts that were fast enough to be ignored compared to the API calls.

[9] Minimum hourly wage in the United States is \$7.5, which can be found at `https://www.worker.gov/pay-for-hours-worked/`. Solving a problem requires 60 seconds on average. Therefore, the price and time cost required to complete a problem are \$0.125 and 60 seconds, respectively.

Plans	Time	Money	Acc.	Utility(User Satis.)
Human	60s	\$0.125	93.20	86.40
CoT Prompting	0.8s	\$0.080	85.04	81.60
Self-Consistency ($\mathbf{N}_{self}$ = 10)	8s	\$0.800	92.49	85.80
MCS ($\mathbf{N}_{MCS}$ = 5, α = 20%)	10.8s	\$0.4925	91.00	84.20
MCS + Self-consistency ($\mathbf{N}_{MCS}$ = 5, α = 20%)	10.8s	\$0.4925	93.50	88.80
MCS ($\mathbf{N}_{MCS}$ = 5, α = 40%)	16.8s	\$0.505	92.51	85.60
MCS + Self-consistency ($\mathbf{N}_{MCS}$ = 5, α = 40%)	16.8s	\$0.505	94.09	90.80

Table 17: Analysis of cost and utility for SingleEq. MCS + Self-consistency generally outperforms other methods with higher utility and acceptable cost. $\mathbf{N}$.: # sampled rationale. α: DE threshold. Acc.: Accuracy. User Satis.: User Satisfaction.

5. *MCS + Self-consistency ($\mathbf{N}_{MCS}$ = 5, α = 20%)*: A MCS + Self-consistency ($\mathbf{N}_{MCS}$ = 5, α = 20%) plan that requires to sample from the decoder $\mathbf{N}_{MCS}$ times, which costs the same as the MCS ($\mathbf{N}_{MCS}$ = 5, α = 20%) plan.

6. *MCS ($\mathbf{N}_{MCS}$ = 5, α = 40%)*: MCS samples from LLM decoder $\mathbf{N}_{MCS}$ times and uses top α as threshold, requiring $(\mathbf{N}_{MCS}+1)*\mathbf{p}_{llm}+\alpha*\mathbf{p}_{MCS}$ and $(\mathbf{N}_{MCS}+1)*\mathbf{t}_{llm}+\alpha*\mathbf{t}_{MCS}$ seconds. For $\mathbf{N}_{MCS}=5$, $\alpha=40\%$, the money needed is $(\mathbf{N}_{MCS}+1)*\mathbf{p}_{llm}+\alpha*\mathbf{p}_{MCS} = \$0.08*6+40\%*\$0.0625 = \$0.505$, the time needed is $(\mathbf{N}_{MCS}+1)*\mathbf{t}_{llm}+\alpha*\mathbf{t}_{MCS} = 0.8*6s+40\%*30s = 16.8$ seconds.

7. *MCS + Self-consistency ($\mathbf{N}_{MCS}$ = 5, α = 40%)*: A MCS + Self-consistency ($\mathbf{N}_{MCS}$ = 5, α = 40%) plan that requires to sample from the decoder $\mathbf{N}_{MCS}$ times, which costs the same as the MCS ($\mathbf{N}_{MCS}$ = 5, α = 40%) plan.

The results are shown in Tab. 17. The result shows that MCS +Self-consistency generally outperforms other methods with higher utility (*i.e.*, better user satisfaction) as well as an acceptable cost.

We performed regression analysis on user satisfaction based on LLM and Human and ultimately learned the utility function $\mathbf{u}(\mathbf{x}_{LLM}, \mathbf{x}_{Human}) = \mathbf{x}_{LLM}^{2.05} * (10 * \mathbf{x}_{Human})^{1.94}$, where $\mathbf{x}_{LLM}$ equals to 1 when using LLM to decode one time, and $\mathbf{x}_{Human}$ equals to 10 when solving the problem with only human.

J Related Work

J.1 Human-In-the-Loop System

The human-in-the-Loop system, aiming to achieve what neither humans nor machines can accomplish independently, is defined as a model requiring human interaction (Karwowski, 2006). When the machine cannot solve the problem, or when cost or security considerations require humans to participate, manual intervention is necessary (Wu et al., 2022; Zanzotto, 2019; Mosqueira-Rey et al., 2023). Previous human-in-the-loop systems focus either on adding appropriate tags to data or providing feedback on cases with a certain confidence interval to the machines and thus retrain the model afterward with the labeled data or rewarded cases (Wu et al., 2022; Zanzotto, 2019). The human-in-the-loop system outperforms both standalone AI and humans working alone (Bien et al., 2018).

Recently, LLM-based AI (Artificial Intelligence) systems are developing very quickly, and this trend is expected to expand to the majority of the workforce in the near future (Ouyang et al., 2022; Zhang et al., 2022; Sanh et al., 2021). However, these systems do not always provide satisfactory answers without human intervention, especially mathematical problems. Additionally, in domains such as criminal fact identification and charge predictions, inference should be reasonable and controlled by humans (Custers, 2022) while LLMs are not qualified. Therefore, it is essential to develop a human-in-the-loop prompting-based system that is designed with the ability to collaborate with people. Such a system would make work more efficient and effective. Until recently, few researchers have systematically and quantitatively explored human-in-the-loop prompting-based systems.

Different from ChatGPT's RLHF (Reinforcement Learning from Human Feedback) [10], we take the first step to use human feedback in an online way without access to parameters. Even though it's a preliminary step, this online method could benefit from further refinement and combination with RLHF in future research.

J.2 In-context Learning

Over the past decade, there have been significant advancements in Large Language Models (LLMs) (Ouyang et al., 2022; Zhang et al., 2022; Sanh et al., 2021). These developments have been further accelerated by the introduction of In-Context Learning (ICL) (Kojima et al., 2022). Essentially, LLMs are capable of processing a few training examples and a test instance as its natural language instruction. It then directly decodes the output without requiring any updates to its parameters. LLMs can perform diverse tasks effectively when provided with corresponding instructions (Ouyang et al., 2022; Srivastava et al., 2022; Wei et al., 2022). This presents an opportunity for humans to modify predicted outcomes through natural language instructions, which serve as a flexible and user-friendly interface.

J.3 Chain-of-Thought Prompting

Chain-of-Thought (CoT) prompting enables models to decompose multi-step problems into smaller steps. With CoT, LLMs can solve complex reasoning problems that cannot be solved with standard prompting methods (Wei et al., 2022; Wang et al., 2022). Despite its usefulness, CoT may be prone to errors, which can have a negative impact on the reasoning of the model. Fortunately, most mistakes can be easily interpreted. About half of these mistakes are related to incorrect calculations while the other half are mistakes from flawed reasoning where rationales lack the necessary knowledge (Google Research, 2023). To address this issue, we limit users to modifying, deleting, or adding a single sub-logic as a means of resolving both types of errors. Additionally, we have found that most mistakes can be easily detected and corrected by humans through rationales. Against this background, CoT presents an opportunity for humans to efficiently modify predicted outcomes through sub-logics of rationales.

[10] https://openai.com/blog/chatgpt.