How Much Do Prompting Methods Help LLMs on Quantitative Reasoning with Irrelevant Information?

Seok Hwan Song
Iowa State University
Department of Computer Science
Ames, Iowa, USA
song92@iastate.edu

Wallapak Tavanapong
Iowa State University
Department of Computer Science
Ames, Iowa, USA
tavanapo@iastate.edu

Abstract

Real-world quantitative reasoning problems are complex, often including **extra information irrelevant to the question** (or "**IR noise**" for short). State-of-the-art (SOTA) prompting methods have increased the Large Language Model's ability for quantitative reasoning on grade-school Math Word Problems (MWPs). To assess how well these SOTA methods handle IR noise, we constructed four new datasets with IR noise, each consisting of 300 problems from each of the four public datasets: MAWPS, ASDiv, SVAMP, and GSM8K, with added IR noise. We called the collection of these new datasets "MPN"—Math Word Problems with IR Noise. We evaluated SOTA prompting methods using MPN. We propose Noise Reduction Prompting (NRP) and its variant (NRP$^+$) to reduce the impact of IR noise. **Findings:** Our IR noise significantly degrades the performance of Chain-of-Thought (CoT) Prompting on three different backend models: ChatGPT (gpt-3.5-turbo-0613), PaLM2, and Llama3-8B-instruct. Among them, ChatGPT offers the best accuracy on MPN with and without IR noise. With IR noise, performances of CoT, Least-To-Most Prompting, Progressive-Hint Prompting, and Program-aided Language Models with ChatGPT were significantly impacted, each with an average accuracy drop of above 12%. NRP is least impacted by the noise, with a drop in average accuracy to only around 1.9%. Our NRP$^+$ and NRP perform comparably in the presence of IR noise.

CCS Concepts

• **Computing methodologies → Information extraction**.

Keywords

Prompt Engineering, Large Language Model, Trustworthy Information Extraction, Quantitative Reasoning, Math Word Problem

ACM Reference Format:
Seok Hwan Song and Wallapak Tavanapong. 2024. How Much Do Prompting Methods Help LLMs on Quantitative Reasoning with Irrelevant Information?. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24), October 21–25, 2024, Boise, ID, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3627673.3679840

Table 1: Examples of our proposed noises for quantitative reasoning. Subjects and entities are highlighted in blue and pink, respectively. Noise is shown in yellow.

Original Math Word Problem in the SVAMP dataset:
Body: Claire makes a 3 egg omelet every morning for breakfast.
Question: How many dozens of eggs will she eat in 4 weeks?
Noise disconnected from any sentence:
Body: Claire makes a 3 egg omelet every morning for breakfast.
There are 15 TVs.
Question: How many dozens of eggs will she eat in 4 weeks?
Noise connected to Body sentence:
Body: Claire makes a 3 egg omelet every morning for breakfast.
Scott makes a 23 egg omelet every morning for breakfast.
Question: How many dozens of eggs will she eat in 4 weeks?
Noise connected to Question sentence:
Body: Claire makes a 3 egg omelet every morning for breakfast.
Scott will eat 2 dozens of eggs in 4 weeks.
Question: How many dozens of eggs will she eat in 4 weeks?

Quantitative reasoning is an essential and challenging task, requiring numerical computation and reasoning skills [31]. Previous works attempted to assess and improve the quantitative reasoning ability of language models using Math Word Problems (MWPs) [5, 17, 26, 28, 30]. Large Language Models (LLMs)' numerical reasoning capability has improved significantly on MWPs [13]. Despite different difficulty levels of MWPs from grade school to Calculus, current LLMs (e.g., ChatGPT) still struggle with correctly solving grade school-level MWPs consistently [22]. Prompting methods [7, 25, 27, 32, 33] have been introduced to improve the quantitative reasoning capability of LLMs.

MWP consists of the body sentence(s) and the question sentence [18]. The body sentence provides numerical information to answer the question in the question sentence. Each sentence has subjects and entities. Quantities are always before entities in the body sentences. Table 1 shows an example where the subject in the body sentence is "Claire." The entity is "egg omelet" with a quantity of 3.

However, real-world data often contains excess information irrelevant to calculating the answer to a given question. We call such excess information "IR noise" to distinguish it from other types of noises, such as converting an integer to a real number. Examples of IR noises in MWPs are entities in the body sentence that are not required to compute the correct answer. Several MWP datasets have low percentages of MWP problems with IR noise, e.g., MAWPS [11] at 6.5%, Math23k [26] at 8.2%, and MathQA [1] at 20.7%. More recent datasets, SVAMP [10] and GSM-IC [20] include more MWP problems with unused entities at 44.5%, and 100%, respectively.

CIKM '24, October 21–25, 2024, Boise, ID, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0436-9/24/10
https://doi.org/10.1145/3627673.3679840

Our contributions are as follows.

- A new dataset collection, Math Word Problems with Noise (MPN). Three types of IR noise with different subjects and entities were added to MWP problems selected from four public MWP datasets. Two of the noise types are related to the body or the question sentences in a given MWP problem. These noise types are unique, not in existing datasets, SVAMP and GSM-IC.
- Extensive experiments on MPN and derived GSM-IC [20] datasets to evaluate the impact of IR noise on Chain of Thought (CoT) Prompting [27], Self-Consistency (SC) [25], Least-to-Most Prompting (LTM) [33], Program-aided Language models (PAL) [7], and Progressive-Hint Prompting (PHP) [32]. We tested these methods using ChatGPT (gpt-3.5-turbo-0613), PaLM-2-text-bison-001, and Llama3-8B-instruct as the backend LLM. On MPN, CoT has an absolute average accuracy drop of around 14.4% with ChatGPT as the backend and higher with PaLM2 as the backend. With ChatGPT, LTM, PHP, and PAL have an average accuracy drop of 12.2%, 7.2%, and 24.1%, respectively. Moreover, the noises in our MPN dataset collection degrade these LLMs' reasoning ability more than those in the GSM-IC dataset.
- New prompting methods, Noise Reduction Prompting (NRP) and NRP$^+$, and extensive evaluation of these methods. Both methods prompt the backend LLM to extract noise first. Then, they prompt the LLM to solve MWPs by hinting about the extracted noise. NRP$^+$ differs from NRP in the noise extraction stage. On MPN with IR noise, NRP outperforms self-consistency-based methods and limits the accuracy drop to about 1.9%. However, without any IR noise, NRP performs worse than the other studied state-of-the-art prompting methods showing a performance gap ranging from 3.0% to 10.0% on GSM8K (one dataset of MPN). The noise extraction stage of NRP$^+$ aims to address the drawback of NRP when IR noise is not present. Without IR noise, NRP$^+$ outperforms LTM, PAL, and NRP by 5.0%, 2.0%, and 7.0% in absolute accuracy on GSM8K, respectively, and it underperforms CoT and PHP by only 0.7% and 3.0%, respectively. Furthermore, NRP$^+$ performs slightly worse than NRP when there is noise by around 0.6% on average. NRP$^+$ outperforms the rest of the methods for the datasets with inherent noise, e.g., the derived GSM-IC. We observed that LLM performance improved when several prompts were used to solve a complex task in multiple stages than when only one prompt combining multiple steps was used.

The dataset collection and code are available at https://github.com/ssh1419/MPN.

1 Related work

1.1 Quantitative Reasoning Datasets and Solvers

MWPs have been used for benchmarking numerical reasoning of language models. Diverse datasets, including SVAMP [18], MAWPS [11], and GSM8K [6] were released. Recently, many deep learning methods were proposed: Seq2Seq [4, 14, 23], Seq2Tree [15, 19, 24], and solvers based on LLMs (e.g., GPT [17], LLaMA [21], and PaLM [5]). With LLM-based solvers, few-shot learning through prompting has been outstanding for solving simple question-answering tasks [27]. However, few-shot learning has some limitations [2]. To overcome these limitations, recent prompting methods were proposed, namely Chain of Thought [27], Program of Thoughts Prompting [3], Least-to-Most Prompting [33], Program-aided Language Models [7], and Progressive-Hint Prompting [32].

1.2 MWP Perturbation

Jia and Liang [9] investigated perturbation methods by adding a new sentence to the end of a given body paragraph without changing the meaning of the question and the final answer. Kumar et al. [12] reordered and paraphrased MWP sentences. Patel et al. [18] and Xu et al. [29] suggested changing information, reordering sentences, paraphrasing, and adding irrelevant information. Xu et al. [29] focused on diagnosing numerical capability and the impact of the number format change, however the dataset is not public. Shi et al. [20] introduced the GSM-IC dataset by adding an irrelevant sentence with in-topic and out-topic information using templates on the GSM8K dataset [6]. They provided a prompting method to handle the noise. The prompting method asks the backend LLM to ignore the irrelevant information in the MWP and solve it in one step [20]. Our proposed methods, NRP and NRP$^+$ consist of several steps to enhance the ability to extract noises.

2 Proposed Collection of Math Word Problems with IR Noises (MPN)

The goal of the papers is to investigate the impact of different IR noise on prompting methods on LLM-based solvers to solve MWPs. Subjects and entities are important components of MWPs, but are a small part of MWP perturbations in existing works [18, 29]. Most perturbations were on numerical data. In this paper, we investigate three types of IR noise around subjects and entities. Two of the noise types have not been studied to our knowledge.

2.1 Three Types of IR Noise

Table 1 shows an MWP example and the three types of IR noise we investigated. Recall that MWP consists of the body sentence(s) and the question sentence [18]. In all MWPS, the question sentence is located at the end of the problem after all body sentence(s). Let B and Q be sets of body and question sentences, respectively. The body sentence provides numerical information to answer the question in the question sentence. Each sentence contains subjects (S) and entities (E). Quantities always appear before entities in the body sentences.

B can contain multiple S and E whereas Q contains only the target S and E. T represents the answer according to Q and it needs the target S and E from B for the answer. Table 1 shows an example of an MWP problem. The solution for the problem can be regarded as the output of the function f as defined in Equation 1.

$$f : (B, Q) \rightarrow T \tag{1}$$

Table 2 shows the process to generate the proposed noise. The noise sentence does not affect the final answer to the math word problem. Let S_k and E_k be sets of subjects and entities of sentence k, respectively.

Table 2: Type of proposed noise examples; noise is highlighted in yellow.

Disconnected from Any Sentence (DS)	Connected to Body Sentence (CB)	Connected to Question Sentence (CQ)
(Add a noise sentence where its subject and entity are not in any of the sentences of the original MWP problem.) The noise should not follow any sentence structures.	Claire makes a 3 egg omelet every morning for breakfast. ↓ (Change the subject and/or the entity of the sentence.) ↓ Scott makes 3 cookies every morning for breakfast. ↓ (Change the number in the sentence.) ↓ Scott makes 23 cookies every morning for breakfast.	How many dozens of eggs will she eat in 4 weeks? ↓ (Change the question sentence to a declarative sentence with either the subject or entity different from the question sentence.) ↓ (Change the number in the sentence.) ↓ Scott will eat 24 eggs in 4 weeks.
Body: Claire makes a 3 egg omelet every morning for breakfast. There are 15 TVs. Question: How many dozens of eggs will she eat in 4 weeks?	Body: Claire makes a 3 egg omelet every morning for breakfast. Scott makes a 23 cookies every morning for breakfast. Question: How many dozens of eggs will she eat in 4 weeks?	Body: Claire makes a 3 egg omelet every morning for breakfast. Scott will eat 24 eggs in 4 weeks Question: How many dozens of eggs will she eat in 4 weeks?

Disconnected from any sentence (DS) is a randomly generated sentence where neither the subject nor entity is related to any words in any of the body or question sentences. Furthermore, the generated sentence's structure is unrelated to the body or question sentences. See the noise example in the first column in Table 2.

Let B contain S_b and E_b and Q contain S_q and E_q. We make the noise sentence A contain S_a and E_a where $S_a \neq S_q$ and $S_a \neq S_b$; $E_a \neq E_q$ and $E_a \neq E_b$. We provide six different sentence templates, and the noise sentence follows the randomly selected sentence template R. The generated noises can be regarded as the output of the function g_{ds} as defined in Equation 2.

$$g_{ds} : (S_a, E_a, R) \longrightarrow A \tag{2}$$

Connected to Body Sentence (CB) is a generated sentence with its meaning related to the body sentence. The noise-sentence structure follows the body-sentence structure. The generated subject or entity does not change the final answer.

Let B contain S_b and E_b and Q contain S_q and E_q. We make the noise A contain S_a and E_a, where $S_a \neq S_b$ or $E_a \neq E_b$. If S_a and E_a change the final answer, we manually modify S_a or E_a to keep the original answer. For instance, when the question sentence is "How many apples does he have?", the noise's subject should not be a male's name or the entity should not be apples. If S_a or E_a need to be modified, we manually switch them to a female's name or another object (e.g., "Jennifer" or "peaches"). Let R be a sentence structure of B, and the function g_{cb} only switches the subjects and entities of R to S_a and E_a and returns the generated noise as defined in Equation 3.

$$g_{cb} : (S_a, E_a, R) \longrightarrow A \tag{3}$$

Connected to Question Sentence (CQ) is a generated sentence that follows the template of a declarative version of the question sentence. The goal is for the noise sentence to be semantically close to the question sentence.

Let B contain S_b and E_b and Q contain S_q and E_q. We make the noise A contain S_a and E_a, where $S_a \neq S_q$ or $E_a \neq E_q$. If S_a and E_a change the final answer, we manually modify S_a or E_a as done for CB. The function g_{cq1} returns a template R of a declarative version of Q as defined in Equation 4, and the generated noise can be regarded as the output of the function g_{cq2} as defined in Equation 5.

$$g_{cq1} : (Q) \longrightarrow R \tag{4}$$

$$g_{cq2} : (S_a, E_a, R) \longrightarrow A \tag{5}$$

To the best of our knowledge, the CB and CQ noise types have not been studied in the literature. We hypothesize that they are more complicated for LLMs to solve than the DS noise types. Experimental results in Section 5 confirm our hypothesis.

2.2 Creation of MPN

We generated a new dataset collection, MPN, to study the impact of IR noise. We chose MWP problems from four public datasets ranging from lower-grade elementary school coursework to higher-grade elementary school mathematics: MAWPS [11], ASDiv [16], SVAMP [18], and GSM8K [6], respectively.

For each of these datasets except for GSM8K, we selected 300 MWP questions divided equally into five types of operators (addition, subtraction, multiplication, division, and multi-operators). Since GSM8K only contains multi-operators, we randomly selected 300 MWP questions from the dataset. Out of the 300 questions, we generated DS noise for 100 questions, CB noise for 100 different questions, and CQ noise for the remaining 100 questions. We kept the proportion of the types of operators the same for different types of noise. Therefore, there are 20 questions with each type of operator

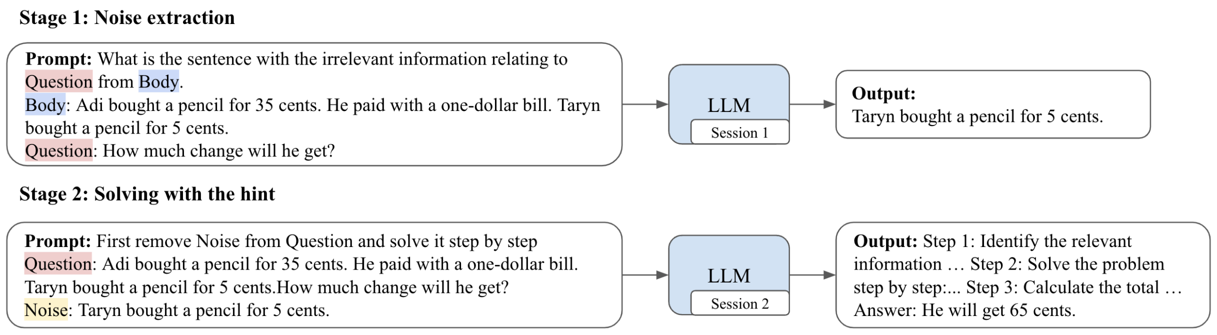

Stage 1: Noise extraction
Prompt: What is the sentence with the irrelevant information relating to Question from Body.
Body: Adi bought a pencil for 35 cents. He paid with a one-dollar bill. Taryn bought a pencil for 5 cents.
Question: How much change will he get?
LLM
Session 1
Output: Taryn bought a pencil for 5 cents.
Stage 2: Solving with the hint
Prompt: First remove Noise from Question and solve it step by step
Question: Adi bought a pencil for 35 cents. He paid with a one-dollar bill. Taryn bought a pencil for 5 cents.How much change will he get?
Noise: Taryn bought a pencil for 5 cents.
LLM
Session 2
Output: Step 1: Identify the relevant information … Step 2: Solve the problem step by step:... Step 3: Calculate the total … Answer: He will get 65 cents.

Figure 1: Proposed NRP on an MWP (ASDiv) with a noise: (1) Extract the noise; (2) Solve the problem by giving useful hints.

for each type of noise. The body and question sentences, equations, and answers are from the original datasets. Pre-processing, such as converting an integer into a real number, was applied to have a consistent sentence format. Different methods were used to generate the noise according to the types of noise. For DS noise, we made six different structure templates of sentences with 12 subjects and 13 entities using a quantity between 1 and 50. For CB and CQ noise, we wrote a Python script with APIs to ChatGPT (gpt-3.5-turbo-0613) to generate the noise. ChatGPT was used to convert interrogative sentences (the question sentences) to corresponding declarative sentences or switch subjects or entities as described in the previous section. The question sentence is placed at the end of the problem following all body sentence(s). Note that our generated noise sentence is added at the end of the body sentences and before the question sentence. We do not place the noise sentence in other locations since they can change the answer to the original MWP question. In our final check, we manually inspected and corrected the generated noise to ensure that the noise did not change the answer to each question.

3 Investigated Prompting Methods

We propose two variants of Noise Reduction Prompting. Both variants prompt a backend LLM to find noise in the first stage. Recall that noise contains irrelevant subjects and/or entities to individual questions. In the second stage, the extracted noise is used in a prompt to the backend LLM as a hint to solve the original reasoning problem. Recent LLMs such as ChatGPT and PaLM2 can solve a problem step by step. Each MWP's body sentences and question sentences are identified prior to the proposed methods being applied.

3.1 Compared Methods

We selected these methods due to their potential to handle IR noises. **Chain of Thought (CoT)** prompts LLMs to solve MWPs step by step, not just the final answer providing exemplars of solutions [27]. CoT significantly improves the reasoning ability of LLMs and is widely used as a baseline prompting method.

Least-To-Most (LTM) prompting solves MWPs in two stages [33]. During the first stage, the backend LLM decomposes MWP into simpler sub-problems, and in the second stage, the LLM solves the generated sub-problems. Shi et al. [20] demonstrated that LTM outperformed other compared prompting methods on their dataset with irrelevant information.

Self Consistency (SC) selects the majority vote of the answers leading to increased accuracy, since LLMs return different answers for the same task at different times [25].

Program-aided Language Models (PAL) generates natural language and programming language together to solve MWPs [7].

Progressive-Hint Prompting (PHP) asks the backend LLM to solve the same MWP question several times. Each time, the backend LLM solves the problem that includes the previously generated answers as a hint for double-checking purposes [32]. PHP achieves significant performance improvement on MWPs [32].

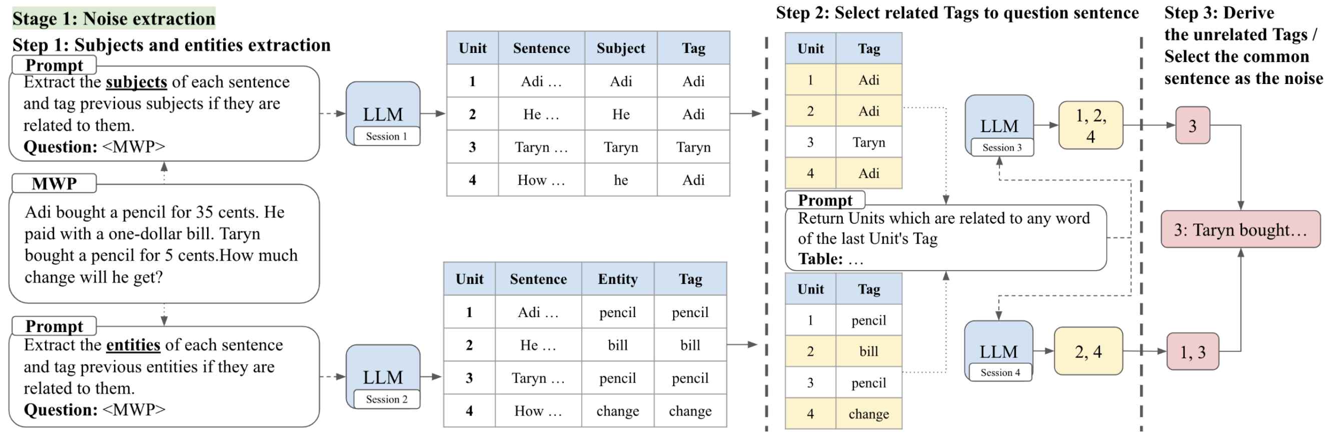
3.2 Proposed Methods

3.2.1 Noise Reduction Prompting (NRP). Figure 1 shows how NRP solves MWP.

(1) **Noise extraction.** Issue a prompt to extract sentences that contain irrelevant information to the question sentence from the body sentence(s). Body sentences and question sentences are provided separately.
(2) **Solving with the hint.** Issue another prompt with two forms of information: (1) the original problem and (2) the hint that marks the extracted irrelevant sentences as "Noise" and an instruction to remove the noise and solve the problem step by step.

3.2.2 Noise Reduction Prompting Plus (NRP$^+$). Asking the backend LLM to extract the noise sentence in one step may not be ideal. We investigate a multi-step method for noise extraction. NRP$^+$ uses prompts and a simple program to extract noise by excluding subjects or entities that are found in the question sentence. Figure 2 shows the prompts and corresponding LLM's output of NRP$^+$ that differs from that of the NRP. The first stage of NRP$^+$ contains three steps, each executed independently in its own session.

- **Step 1: Extract subjects and entities in a compressed format**
  The first step submits two prompts to the LLM, one for extracting a subject and the other for extracting an entity. The prompt contains a predefined instruction and the MWP. The question sentence is placed at the end of the problem. Two different instructions are used for extracting the subject and the entity. See Figure 2. The LLM outputs two tables, one for


Stage 1: Noise extraction
Step 1: Subjects and entities extraction
Prompt
Extract the **subjects** of each sentence and tag previous subjects if they are related to them.
Question: <MWP>
MWP
Adi bought a pencil for 35 cents. He paid with a one-dollar bill. Taryn bought a pencil for 5 cents.How much change will he get?
Prompt
Extract the **entities** of each sentence and tag previous entities if they are related to them.
Question: <MWP>
LLM
Session 1
LLM
Session 2
Step 2: Select related Tags to question sentence
Prompt
Return Units which are related to any word of the last Unit's Tag
Table: …
LLM
Session 3
LLM
Session 4
1, 2, 4
2, 4
Step 3: Derive the unrelated Tags / Select the common sentence as the noise
3
1, 3
3: Taryn bought…

Unit	Sentence	Subject	Tag
1	Adi …	Adi	Adi
2	He …	He	Adi
3	Taryn …	Taryn	Taryn
4	How …	he	Adi

Unit	Sentence	Entity	Tag
1	Adi …	pencil	pencil
2	He …	bill	bill
3	Taryn …	pencil	pencil
4	How …	change	change

Unit	Tag
1	Adi
2	Adi
3	Taryn
4	Adi

Unit	Tag
1	pencil
2	bill
3	pencil
4	change

Figure 2: Stage 1 of the proposed NRP$^+$ on an MWP (ASDiv) with an IR noise. The dotted arrows represent adding contents to prompts (e.g., questions for Step 1 and tables for Step 2). The dashed arrow indicates the input given to the LLM. The solid arrows show the outputs from the LLMs or the simple program. Every LLM session is conducted independently. The yellow color highlights the information related to the question sentence.

each prompt. Each tabular output has four columns: Unit, Sentence, Subject/Entity, depending on the prompt, and Tag. The unit column indicates the order of the sentence in the MWP. The tag column contains the subject or entity of the sentence. The LLM identifies the subject in the previous sentence as a tag if they are related. For example, for the first table in Figure 2, when the second sentence's subject is 'He,' the tag for 'He' is 'Adi,' which is from the tag of the first sentence.

- **Step 2: Select related tags to the question sentence's tag** The LLM is prompted to individually select the unit that contains the related tag to the question sentence's tag for the subject and entity. The prompt includes a predefined instruction and the table from the first step. Only Unit and Tag columns are filtered to discard other information from actual sentences. The LLM returns the units of which tags are related to the tag of the question sentence.
- **Step 3: Derive the unrelated tags and select the common sentences as the noise**
The last step uses a simple program that takes selected tags from the previous step of the subject and entity extraction tasks to find unrelated tags to the question sentence's tag. Next, the program finds the unit(s) representing the commonly selected sentences from the subject and entity extraction tasks. If one of the tasks returns nothing and the other returns only one unit, the unit is selected as the commonly selected sentence. The sentence will be used as a noise in a hint for the second stage as done for NRP. LLMs perform poorly in interpreting contexts that contain negation [8]. We split the process of identifying unrelated tags into two steps: identifying related tags and deriving the unrelated tags.

We propose prompts to extract noises, whereas Shi et al. [20] proposed a prompt to ignore the noise without explicitly extracting the noise.

4 Experimental Setup and Results

We describe the experiment setup, including datasets, backend LLMs, and performance comparison of CoT, NRP, NRP$^+$, and four other prompting methods.

4.1 Experiment Setup

Constructed datasets: (1) MPN is a collection of MWPs selected from four public datasets: SVAMP [18], GSM8K [6], MAWPS [11], and ASDiv [16] with one IR noise added to each MWP as described in Section 2.2. By default, we generated one noise sentence per MWP problem. (2) We derived the GSM-IC-797 dataset from a random sampling of 797 samples from the GSM-IC dataset [20]. The GSM-IC dataset contains 100 base problems from the GSM8K dataset, and each base problem is associated with multiple templates of noise sentences. Each template was used to generate multiple noises with diverse subjects and numbers. GSM-IC-797 covers all 100 base problems and all templates of noise sentences for each base problem.

Backend language models and hyperparameters. Limited resources prevented us from testing many LLMs. We chose ChatGPT (gpt-3.5-turbo-0613) and PaLM2 (text-bison-001) as representatives of large closed-source LLMs. We selected Llama3 8B (llama-8B-instruct) to represent a small open-source LLM for the repeatability of experiments. Hereafter, ChatGPT, PaLM2, and Llama3 8B denote the above versions of ChatGPT, PaLM2, and Llama3 8B, respectively. The default configurations used greedy decoding (i.e., temperature = 0). For self-consistency experiments, the temperature of 0.7 was used per the reference [25]. We conducted additional experiments using ChatGPT with a temperature of 1 to evaluate the impact of hyperparameter choices.

Programming tools and prompting methods. Python 3.10.12 and LLM APIs from Openai and google.generativeai were used. For all compared methods, CoT, LTM, PAL, PHP, NRP, and NRP$^+$, we provided a set of few-shot examples. We used manually designed prompt templates for CoT, LTM, NRP, and NRP$^+$. We used the

Table 3: Accuracy on 300 questions (300 Q) with IR noise (indicated by ✓) added to the original MWP problems and without ones. The column headers DS, CB, and CQ indicate the noise types when the noise was added. AV. indicates average accuracy across the row within each dataset. Underlined accuracy is the highest in each column on MWPs without IR noise. Accuracy in bold is the highest in each column on MWPs with IR noise.

Model	Method	Noise	MPN dataset collection constructed from four different datasets															
			MAWPS (300 Q)				ASDiv (300 Q)				SVAMP (300 Q)				GSM8K (300 Q)			
			DS	CB	CQ	AV.	DS	CB	CQ	AV.	DS	CB	CQ	AV.	DS	CB	CQ	AV.
ChatGPT	CoT		88.0	90.0	92.0	90.0	92.0	92.0	94.0	92.7	91.0	79.0	88.0	86.0	79.0	79.0	76.0	78.0
		✓	**85.0**	**70.0**	**70.0**	**75.0**	**84.0**	**82.0**	**69.0**	**78.3**	**84.0**	**67.0**	**63.0**	**71.3**	**70.0**	**63.0**	**61.0**	**64.6**
PaLM2	CoT		85.0	86.0	88.0	86.3	84.0	84.0	86.0	84.7	72.0	62.0	67.0	67.0	67.0	69.0	58.0	64.7
		✓	77.0	63.0	45.0	61.7	79.0	57.0	59.0	65.0	64.0	46.0	33.0	47.7	55.0	57.0	36.0	49.3
Llama3 8B	CoT		73.0	70.0	83.0	75.3	79.0	85.0	80.0	81.3	89.0	74.0	82.0	81.7	63.0	69.0	61.0	64.3
		✓	68.0	63.0	57.0	62.7	71.0	68.0	**69.0**	69.3	83.0	63.0	57.0	67.7	57.0	55.0	38.0	50.0

prompts and code provided by the original authors of PAL and PHP for their methods. For SC experiments, we sampled 10 responses. CoT was used as the baseline prompting method since it was commonly used as the baseline in several studies [7, 20, 25].

Performance metrics. Accuracy is a commonly used performance metric. It is defined as the percentage of the number of correctly answered questions to the total number of questions. For each question, we ran the program (MWP solver) once and compared the final numeric answer from the backend LLM's output with the ground truth. We report the accuracy for each dataset using all the questions in the dataset. Performance comparisons among compared methods and settings are discussed in terms of absolute differences in accuracy.

4.2 Main Results on MPN

Table 3 shows that ChatGPT is best at solving the original MWPs, offering average accuracy ranging from 78.0% to 92.7%. PaLM2 performs better than Llama3-8B on MAWPS, ASDiv, and GSM8K, while Llama3-8B outperforms PaLM2 on SVAMP.

4.2.1 Which LLM is better at solving MWPs with the IR noise? Table 3 presents the answers. All three LLMs are severely impacted by the noise. ChatGPT's absolute average accuracy drops range from 13.4% to 15.0% across the datasets. Llama3-8B's accuracy drops vary from 12.0% to 14.3%. PaLM2 suffers the highest accuracy drops, ranging from 15.4% to 24.6%. Nevertheless, ChatGPT still offers the best average accuracy, followed by Llama3-8B. Hence, we chose ChatGPT as the baseline LLM for the rest of the studies.

4.2.2 Impact of IR noise on ChatGPT with CoT. We demonstrate the various impacts of IR noise on the chosen LLM, ChatGPT.

Which noise type does ChatGPT find most difficult to handle? Table 3 shows that CQ noise is the most challenging, causing ChatGPT with CoT to drop between 15.0% and 25.0% in accuracy with an average drop of 22.0%. The CB noises lead to a 14.5% accuracy drop on average. The DS noise is the least effective. MWPs selected from MAWPS were the least sensitive, with around a 3.0% accuracy drop, while the accuracy drop in the other datasets is around 8.0%.

Categories of ChatGPT's incorrect answers. Figure 3(top) shows categories of ChatGPT's incorrect answers on the MPN collection of datasets. Two of the categories ChatGPT claimed due to

Category	Description
Partially correct	Correct results in some intermediate steps
Wrong	Completely wrong answer
No solution	Unable to solve the MWP
Wrong answers with noise	Wrong answer with noise used in intermediate steps
No solution with noise	Unable to solve the MWP due to the noise

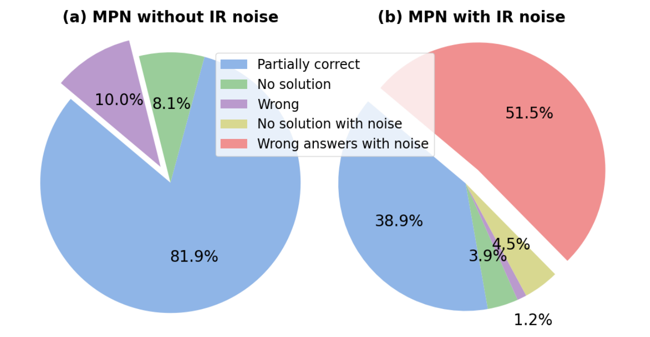


Figure 3: Percent of ChatGPT's incorrect answers on MPN (a) without IR noise and (b) with IR noise.

the noise in MWPs. The category *Wrong answer with noise* includes cases ChatGPT returns an incorrect answer due to the noise. The category *No solution with noise* includes cases ChatGPT says it cannot solve the problem because of the noise. Figure 3(a) shows that, without the IR noise, 81.9% of all incorrect answers are partially correct. With IR noise, the noise causes 56.0% of the incorrect answers, of which 4.5% ChatGPT cannot give solutions, and 51.5% ChatGPT gives incorrect solutions.

Impact of IR noises by operator types on CoT with ChatGPT. GSM8K was excluded from this study since each of its math word problems has more than one operator type. Figure 4 indicates several notable trends in the datasets. The biggest difference in MAWPS is 21.7%, related to the multi-operators problems with the IR noise, whereas the smallest drop is about 10.0% on the addition problems. ASDiv has a significant accuracy reduction of 25.0% on the addition problems with the IR noise, with the smallest gap being about 6.7% on the division problems. The IR noise on SVAMP

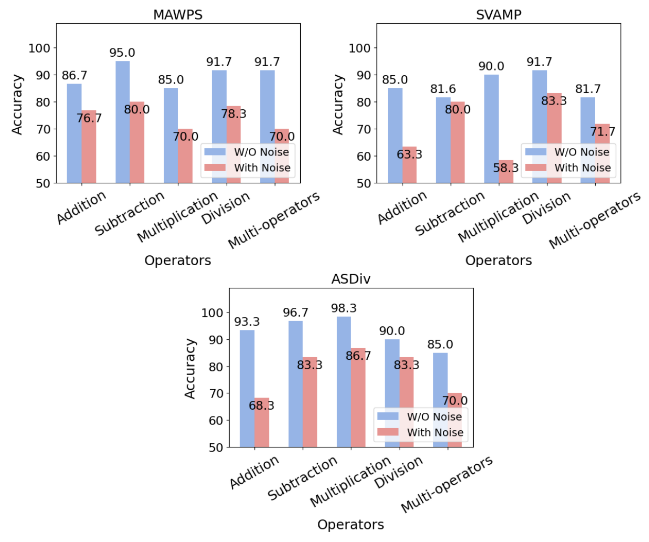


Figure 4: Accuracy of CoT with ChatGPT by operator types in the MPN collection excluding GSM8K.

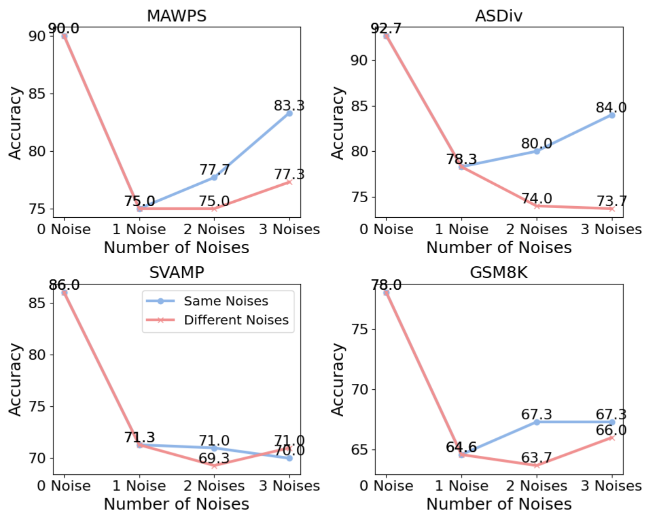


Figure 5: Average accuracy of CoT with ChatGPT on MPN when using the same noise sentences and different noise sentences per MWP is shown in blue and pink, respectively.

causes the largest accuracy drop of 31.7% on the multiplication problems and the smallest gap of 1.6% on the subtraction problems. The accuracy gaps are very diverse depending on the operator types.

Impact of multiple IR noise sentences on CoT with ChatGPT. We tested up to three IR noise sentences per MWP problem. The noise sentences may be exactly the same or have different elements (subjects, entities, and numbers) in the same sentence structure. Figure 5 demonstrates that multiple different noises are more effective at reducing accuracy than multiple identical noises. For SVAMP and GSM8K, two different noise sentences are the most effective, causing accuracy drops by 16.7% and 14.3%, respectively. The highest accuracy drops for ASDiv are due to three different noise sentences. For MAWPS, one and two different noise sentences caused the largest accuracy reduction. Except SVAMP, multiple identical noises are less effective than a single noise sentence.

4.3 Effectiveness of Prompting Methods with ChatGPT

Table 4 shows the effectiveness of four state-of-the-art prompting methods and the proposed NRP and NRP$^+$. Self-consistency (SC) was used with CoT and LTM only. We have discussed the impact of our IR noise on CoT with ChatGPT previously. Next, we discussed the rest of the prompting methods. Figure 6 shows the accuracy difference by a prompting method on MWPs without IR noise and the same MWPs but with added IR noise.

- **Least-To-Most (LTM)** is impacted by the IR noise, with the absolute average accuracy drop ranging from 7.3% to 16.7% across the datasets. The CQ noise type is the most effective, causing between 11.0% and 26.0% drops. The DS noise is the least effective. Among the datasets, as Figure 6 shows, SVAMP is the most sensitive with around 16.7% accuracy drop, while GSM8K is the least sensitive with around 7.3% accuracy drop.
- **Self Consistency (SC)** helps both CoT and LTM be better at handling the IR noise than CoT and LTM without SC. CoT+SC offers higher accuracy than CoT in the range of 1% to 10.4% across the four datasets, with 4.3% increases on average. LTM+SC offers more accuracy improvement over LTM, ranging from 2.7% to 12.1% with 7.6% increases on average. Therefore, SC improves the performance with noise, and SC helps LTM more than CoT.
- **Program-aided Language Models (PAL)** suffers a significant drop in accuracy under IR noise by ranging from 19.6% to 27.3%. The IR noise on SVAMP causes the largest drop and CQ is the most effective type among other types.
- **Progressive-Hint Prompting (PHP)** performs great without IR noise, ranging from 80.3% to 93.7%. However, the accuracy drops significantly, ranging from 13.0% to 22.0%. The IR noise on SVAMP causes the largest drop, and CQ is the most effective noise type among other types.
- **The proposed NRP** handles the noise quite well, as evidenced by smaller accuracy drops compared to when there was no noise, i.e., 2.4% (MAWPS), 1.0% (ASDiv), 5.4% (SVAMP). Under noise, NRP offers increased accuracy over CoT from 7.0% to 14.3%. However, NRP performs worse than CoT on ASDiv and GSM8K without noise. Because the first step of NRP often returns some sentences in the absence of noise. There are no significant differences between the effects of the types of noise. NRP offers the worst improvement over CoT on GSM8K, but the improvement of 7.0% increase in accuracy is still significant. On the rest of the datasets, NRP offers more than a 10.0% absolute increase in accuracy over CoT. The differences in accuracy between the types of noise are not huge. CQ noise shows the biggest influence.
- **The proposed NRP$^+$** handles the noise as well as NRP as evidenced by very small accuracy drops compared to NRP,

Table 4: Accuracy of different prompting methods with ChatGPT on 300 questions (300 Q) with IR noise (indicated by ✓) added to the original MWPs and without ones. The column headers DS, CB, and CQ indicate DS, CB, and CQ noise, respectively when noise was added. AV. indicates average accuracy across the row within each dataset. Underlined accuracy is the highest in each column on MWPs without IR noise. Accuracy in bold is the highest in each column on MWPs with IR noise.

Method	Noise	MPN dataset collection constructed from four different datasets															
		MAWPS (300 Q)				ASDiv (300 Q)				SVAMP (300 Q)				GSM8K (300 Q)			
		DS	CB	CQ	AV.	DS	CB	CQ	AV.	DS	CB	CQ	AV.	DS	CB	CQ	AV.
CoT		88.0	90.0	92.0	90.0	92.0	92.0	94.0	92.7	91.0	79.0	88.0	86.0	79.0	79.0	76.0	78.0
	✓	85.0	70.0	70.0	75.0	84.0	82.0	69.0	78.3	84.0	67.0	63.0	71.3	70.0	63.0	61.0	64.6
CoT+SC	✓	86.0	74.0	73.0	77.7	88.0	81.0	69.0	79.3	90.0	69.0	60.0	73.0	**81.0**	**78.0**	66.0	75.0
LTM		91.0	91.0	92.0	91.3	90.0	89.0	88.0	89.0	87.0	79.0	88.0	84.7	76.0	73.0	71.0	73.3
	✓	87.0	71.0	74.0	77.3	87.0	75.0	73.0	78.3	82.0	60.0	62.0	68.0	72.0	66.0	60.0	66.0
LTM+SC	✓	89.0	75.0	76.0	80.0	**92.0**	84.0	75.0	83.7	**95.0**	74.5	72.5	80.1	79.0	75.0	**75.0**	**76.3**
PAL		87.0	86.0	92.0	88.3	87.0	94.0	91.0	90.7	89.0	74.0	85.0	82.7	67.0	82.0	77.0	75.3
	✓	80.0	58.0	45.0	61.0	81.0	72.0	46.0	66.3	74.0	55.0	43.0	57.3	57.0	55.0	55.0	55.7
PHP		87.0	94.0	93.0	91.3	93.0	96.0	92.0	93.7	83.0	81.0	82.0	82.0	79.0	82.0	80.0	80.3
	✓	79.0	77.0	66.0	74.0	85.0	78.0	69.0	77.3	68.0	61.0	51.0	60.0	76.0	69.0	57.0	67.3
NRP		91.0	89.0	92.0	90.7	86.0	92.0	90.0	89.3	92.0	84.0	87.0	87.7	70.0	73.0	68.0	70.3
	✓	88.0	**89.0**	**88.0**	**88.3**	91.0	**90.0**	**84.0**	**88.3**	**95.0**	76.0	**76.0**	**82.3**	76.0	70.0	69.0	71.6
NRP$^+$		94.0	96.0	93.0	94.3	90.0	93.0	90.0	91.0	91.0	82.0	83.0	85.3	79.0	77.0	76.0	77.3
	✓	**92.0**	85.0	87.0	88.0	91.0	89.0	81.0	87.0	92.0	**80.0**	75.0	**82.3**	78.0	70.0	65.0	71.0

i.e., 0.3% (MAWPS), 1.3% (ASDiv), and 0.6% (GSM8K). NRP$^+$ performs better on some types of noise than NRP, such as DS and CB. NRP$^+$ performs better than NRP on MAWPS, ASDiv, and GSM8K without noise. Moreover, on MAWPS, NRP$^+$ outperforms CoT by 4.3%, and there were very small accuracy drops compared to the CoT on ASDiv, SVAMP, and GSM8K, i.e., 1.7% (ASDiv), 0.7% (SVAMP), and 0.7% (GSM8K) without noise, respectively. Therefore, NRP$^+$ performs great on both MWPs with and without IR noise.

LTM performs slightly better than CoT on MAWPS, ASDiv, and GSM8K with noise. The CQ noise type causes the most drop in accuracy for all the methods. GSM8K contains the most difficult MWPs, resulting in the lowest accuracy. Table 4 also reveals that the variability in the performance of these methods is higher with noises.

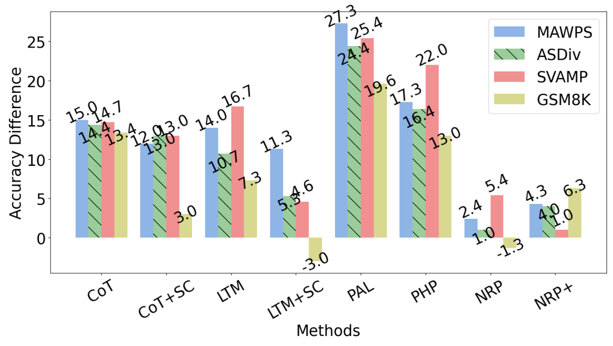


Figure 6: The negative number differences (by LTM+SC and NRP) indicate the cases when the prompting methods on MWPs with IR noise perform better than without IR noise.

Figure 6 shows that with IR noise, NRP performs the best on MAWPS and ASDiv, NRP$^+$ and NRP perform the best on SVAMP, LTM+SC performs the best on GSM8K. NRP and NRP$^+$ successfully handle the noise, experiencing drops of no more than 5.4% and 6.3%, respectively. While PAL and PHP suffer a significant drop in accuracy under noise by ranging from 19.6% to 27.3% and ranging from 13.0% to 22.0%, respectively. Especially, PHP performs great without noise, but it performs even worse than CoT with noise. SVAMP is the most sensitive dataset across most prompting methods (LTM, PAL, PHP, and NRP), showing the largest accuracy drop.

Some datasets of MPN originally have body sentences containing irrelevant information. Specifically, MAWPS, ASDiv, SVAMP, and GSM8K of MPN have 44.7%, 2.7%, 50.0%, and 0.3% of their MWPs with the original irrelevant information. Therefore, Table 4 shows that NRP and NRP$^+$ even perform better on MAWPS and SVAMP without IR noise than other prompting methods, while PHP performs the best on ASDiv and GSM8K.

5 Performance of NRP and NRP$^+$ with ChatGPT on GSM-IC-797

How well do our NRP variants help ChatGPT handle MWPs with other irrelevant information? We answer this question using GSM-IC-797. This dataset has 797 problems with noise from GSIM-IC proposed by Shi et al. [20], which were generated from 100 base problems. The authors reported that LTM with their instructed prompting for the backend LLM to ignore irrelevant information, performs the best on GSM-IC. Table 5 shows the performance of our NRP variants, CoT, and LTM without and with instructed prompting [20]. CoT and LTM with instructed prompting give better accuracy on average. Therefore, when describing the performance comparison with CoT and LTM, we refer to the versions with instructed prompting [20] shown in parentheses in Table 5. On 100 base problems without noise, NRP$^+$ performs better than CoT and performs similarly to LTM. On GSM-IC-797, NRP outperforms CoT and LTM by 1.4% and 3.5% on average, respectively. NRP$^+$ outperforms CoT and LTM by 0.9% and 3.0% on average, respectively.

To examine whether our CQ noises are more challenging than Shi et al.'s irrelevant information [20], we created GSM-IC-CQ. A CQ noise is added for each of the 100 base problems from GSM-IC-797. On GSM-IC-CQ, the average accuracy drops of CoT and LTM are 12.6% and 16.7%, respectively. NRP$^+$ greatly outperforms the other methods by 6.5% to 9.4%. Our noise is more difficult than those in the GSM-IC dataset since they cause more accuracy drops. NRP and NRP$^+$ outperform LTM.

Table 5: Accuracy of different prompting methods on GSM-IC-797 with ChatGPT. Numbers inside the parentheses are accuracies from the new experiments with ChatGPT using the instructed prompting [20]. Problems requiring 2 step reasoning are denoted as "2 Steps" where "> 2 Steps" indicates problems requiring more than 2 reasoning steps.

Dataset		CoT	LTM	NRP	NRP$^+$
w/o noise (GSM-IC-797 -base problems) (100Q)	2 Steps	91.6 (88.5)	90.8 (92.9)	**88.1**	**93.3**
	>2 Steps	85.3 (93.4)	91.8 (91.5)	**86.5**	**90.0**
	AVG	88.5 (91.0)	91.3 (92.2)	**87.3**	**92.0**
w/ noise (GSM-IC-797) (797Q)	2 Steps	85.6 (88.1)	85.8 (87.7)	**90.6**	**90.2**
	>2 Steps	82.1 (87.5)	83.1 (83.7)	**87.8**	**87.2**
	AVG	83.9 (87.8)	84.5 (85.7)	**89.2**	**88.7**
w/ noise (GSM-IC-CQ) (100Q)	2 Steps	81.7	76.7	**80.0**	**88.3**
	>2 Steps	70.0	72.5	**75.0**	**77.5**
	AVG	75.9	74.6	**77.5**	**84.0**

6 Ablation Study and Sensitivity Analysis

We conducted two studies to investigate the impact of each stage of NRP and NRP$^+$ and one study to evaluate the impact of temperature hyperparameter values on prompting methods with ChatGPT.

The first study investigated the effectiveness of the noise extraction stage alone. We executed only the noise extraction stage and compared the extracted sentence to the ground truth IR noise sentence. Table 6 shows that ChatGPT achieves 92.3% average accuracy with NRP and 77.2% with NRP$^+$.

Table 6: Accuracy using only the noise extraction stage of NRP and NRP$^+$ with ChatGPT on MPN with IR noises

	MPN dataset collection				
	MAWPS	ASDiv	SVAMP	GSM8K	AVG
NRP	91.3	91.7	93.0	93.0	92.3
NRP$^+$	77.0	77.5	79.8	74.3	77.2

Table 7: Accuracy of NRP without the noise extraction stage, NRP, and NRP$^+$

	MPN dataset collection				
	MAWPS	ASDiv	SVAMP	GSM8K	AVG
NRP w/o the result of Noise Extraction	84.0	83.7	77.0	67.3	78.0
NRP	88.3	88.3	82.3	71.6	82.6
NRP$^+$	88.0	87.0	82.3	71.0	82.1

In the second study, we executed only Stage 2 of NRP to solve MWPs. Since Stage 1 was not run, there was no extracted noise. Stage 2 prompt became a static prompt that reads, "First remove Noise from Question and solve it step by step", the question, and "Noise:." NRP without the noise hint offers worse average accuracy than NRP by 4.6%. This noise hint is indeed important.

The third study investigated the effect of hyperparameter values of ChatGPT on the average accuracy of CoT and our NRP variants.

Table 8: Impact of temperature hyperparameter values on prompting methods with ChatGPT

Temperature Parameter	Noise (✓)	Prompting Methods		
		CoT	NRP	NRP$^+$
Temp = 0		86.4	84.5	87.0
	✓	71.8	82.6	82.0
Temp = 1		82.1	82.7	84.9
	✓	74.2	79.7	78.7

Table 8 shows interesting results. Without the IR noise, CoT with a temperature of 0 gives better accuracy. On the contrary, with IR noise, CoT with a temperature of 1 gives better accuracy. NRP and NRP$^+$ with a temperature of 0 perform better with and without noises than with a temperature of 1.

7 Conclusions and Future Work

We present an extensive evaluation of the effectiveness of prompting methods in assisting LLMs in solving quantitative reasoning problems with irrelevant information. All investigated LLMs struggle with irrelevant information, and the errors cannot be completely handled with the latest prompting methods that perform well without noise. The LLMs are most impacted by the irrelevant information where the structure of the added noise sentence follows that of the question sentence. Our new prompting methods, NRP variants, can reduce the negative impact of irrelevant information. The noise extraction methods can be extended to filter out irrelevant information for Retrieval-Augmented Generation (RAG) systems. Possible future work includes solving more complex quantitative reasoning problems with complex noises.

8 Acknowledgement

This work is partially supported by the NSF Grant No. 2152117. Findings, opinions, and conclusions expressed in this paper by the authors do not necessarily reflect the view of the funding agency.

References

[1] Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. MathQA: Towards Interpretable Math Word Problem Solving with Operation-Based Formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*. 2357–2367.

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

[3] Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks. *Transactions on Machine Learning Research* (2023).

[4] Ting-Rui Chiang and Yun-Nung Chen. 2019. Semantically-Aligned Equation Generation for Solving and Reasoning Math Word Problems. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*. 2656–2668.

[5] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311* (2022).

[6] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168* (2021). https://arxiv.org/pdf/2110.14168.pdf

[7] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. In *International Conference on Machine Learning*. PMLR, 10764–10799.

[8] Iker García-Ferrero, Begoña Altuna, Javier Álvez, Itziar Gonzalez-Dios, and German Rigau. 2023. This is not a Dataset: A Large Negation Benchmark to Challenge Large Language Models. *arXiv preprint arXiv:2310.15941* (2023).

[9] Robin Jia and Percy Liang. 2017. Adversarial Examples for Evaluating Reading Comprehension Systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 2021–2031.

[10] Zhanming Jie, Jierui Li, and Wei Lu. 2022. Learning to Reason Deductively: Math Word Problem Solving as Complex Relation Extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1)*. 5944–5955.

[11] Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. MAWPS: A math word problem repository. In *Proceedings of the 2016 conference of the north american chapter of the association for computational linguistics: human language technologies*. 1152–1157.

[12] Vivek Kumar, Rishabh Maheshwary, and Vikram Pudi. 2021. Adversarial Examples for Evaluating Math Word Problem Solvers. In *Findings of the Association for Computational Linguistics: EMNLP 2021*. 2705–2712.

[13] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. 2022. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems* 35 (2022), 3843–3857.

[14] Jierui Li, Lei Wang, Jipeng Zhang, Yan Wang, Bing Tian Dai, and Dongxiang Zhang. 2019. Modeling intra-relation in math word problems with different functional multi-head attentions. In *Proceedings of the 57th annual meeting of the association for computational linguistics*. 6162–6167.

[15] Qianying Liu, Wenyv Guan, Sujian Li, and Daisuke Kawahara. 2019. Tree-structured decoding for solving math word problems. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*. 2370–2379.

[16] Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2020. A Diverse Corpus for Evaluating and Developing English Math Word Problem Solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 975–984.

[17] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]

[18] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP Models really able to Solve Simple Math Word Problems?. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2080–2094.

[19] Jinghui Qin, Lihui Lin, Xiaodan Liang, Rumin Zhang, and Liang Lin. 2020. Semantically-Aligned Universal Tree-Structured Solver for Math Word Problems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (Eds.). Association for Computational Linguistics, Online, 3780–3789.

[20] Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*. PMLR, 31210–31227.

[21] 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* (2023). https://arxiv.org/pdf/2302.13971.pdf

[22] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

[23] Lei Wang, Yan Wang, Deng Cai, Dongxiang Zhang, and Xiaojiang Liu. 2018. Translating a Math Word Problem to a Expression Tree. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 1064–1069.

[24] Lei Wang, Dongxiang Zhang, Jipeng Zhang, Xing Xu, Lianli Gao, Bing Tian Dai, and Heng Tao Shen. 2019. Template-based math word problem solvers with recursive neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 7144–7151.

[25] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *The Eleventh International Conference on Learning Representations*.

[26] Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 conference on empirical methods in natural language processing*. 845–854. https://aclanthology.org/D17-1088.pdf

[27] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems* 35 (2022), 24824–24837.

[28] Zhipeng Xie and Shichao Sun. 2019. A Goal-Driven Tree-Structured Neural Model for Math Word Problems.. In *Ijcai*. 5299–5305.

[29] Jialiang Xu, Mengyu Zhou, Xinyi He, Shi Han, and Dongmei Zhang. 2022. Towards Robust Numerical Question Answering: Diagnosing Numerical Capabilities of NLP Systems. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 7950–7966.

[30] Jipeng Zhang, Lei Wang, Roy Ka-Wei Lee, Yi Bin, Yan Wang, Jie Shao, and Ee-Peng Lim. 2020. Graph-to-tree learning for solving math word problems. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 3928–3937.

[31] Wenqi Zhang, Yongliang Shen, Yanna Ma, Xiaoxia Cheng, Zeqi Tan, Qingpeng Nong, and Weiming Lu. 2022. Multi-View Reasoning: Consistent Contrastive Learning for Math Word Problem. In *Findings of the Association for Computational Linguistics: EMNLP 2022*. 1103–1116.

[32] Chuanyang Zheng, Zhengying Liu, Enze Xie, Zhenguo Li, and Yu Li. 2023. Progressive-hint prompting improves reasoning in large language models. *arXiv preprint arXiv:2304.09797* (2023).

[33] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, et al. 2022. Least-to-Most Prompting Enables Complex Reasoning in Large Language Models. In *The Eleventh International Conference on Learning Representations*.