1 Introduction

1.1 Motivation

1.2 Technical Background

1.2.1 Large Language Models

Large language models (LLMs) are essentially highly sophisticated *next word predictors*. Given an input message (or *prompt*), an LLM generates a likely next word in the sequence. Passing the original prompt plus the generated word as input back into the same LLM allows us to generate a second word. This can be repeated to generate a third word and so on, to generate an entire response.

Although conceptually similar, LLMs in fact use *tokens*, not words. Tokens can be whole words, parts of words, or letters, and may be combined with punctuation. The set of tokens in use by an LLM is known as its *vocabulary*, whose set of IDs is denoted by $\mathcal{V} = \{0, 1, \ldots, V - 1\}$ and V is the vocabulary size.

The input prompt, in text, is first broken up and encoded into a sequence of token IDs. This sequence can be written as:

$$\mathbf{w}_{<t} = [w_1, w_2, \ldots, w_{t-1}]$$

Where each token ID $w_i \in \mathcal{V}$. The LLM then generates a probability distribution over possible next tokens given this input sequence:

$$\mathrm{P}(w_t \mid \mathbf{w}_{<t})$$

This probability distribution can then be sampled from to generate the next token in the sequence.

This abstraction of an LLM is sufficient for describing most of the methods in this paper.

1.2.2 Hallucination

1.2.3 Retrieval-augmented Generation

1.2.4 Contrastive Decoding

1.2.5 Chain-of-Thought

2 Additive Context-aware Decoding to Address Issues with Contrastive Decoding

2.1 Method

2.1.1 Context-Aware Decoding (CAD)

Context-Aware Decoding (CAD), introduced by Shi et al. (2024), applies a contrastive decoding strategy to amplify the distinction between output probabilities when a model is used with and without access to context. This method is particularly effective in overriding a model's prior knowledge when it conflicts with the provided contextual information. Formally, let $\mathbf{x}$ denote a question or query, $\mathbf{c}$ a useful context, and $\mathbf{y}< t$ the sequence of previously generated tokens. The output distribution under CAD is defined as:

$$\tilde{\mathrm{P}}_{\mathrm{CAD}}(y_t) \propto \mathrm{P}(y_t \mid \mathbf{c}, \mathbf{x}, \mathbf{y}_{<t}) \left(\frac{\mathrm{P}(y_t \mid \mathbf{c}, \mathbf{x}, \mathbf{y}_{<t})}{\mathrm{P}(y_t \mid \mathbf{x}, \mathbf{y}_{<t})} \right)^{\alpha}$$

That is, the token distribution with context is scaled by the pointwise ratio between the distributions with and without context, raised to the CAD parameter $\alpha \in \mathbb{R}$.

2.1.2 Additive CAD

As will be motivated by several case studies in section 2.3.4, multiplying by a ratio of probability distributions can sometimes cause extremely small probabilities—when paired with a large ratio—to be disproportionately amplified, occasionally resulting in nonsensical outputs. To address this, we introduce *Additive CAD*.

Using the same notation as above, the output distribution under Additive CAD is given by:

$$\tilde{\mathrm{P}}_{\mathrm{AddCAD}}(y_t) \propto \mathrm{P}(y_t \mid \mathbf{c}, \mathbf{x}, \mathbf{y}_{<t}) + \gamma \left(\mathrm{P}(y_t \mid \mathbf{c}, \mathbf{x}, \mathbf{y}_{<t}) - \mathrm{P}(y_t \mid \mathbf{x}, \mathbf{y}_{<t}) \right)$$

That is, the context-informed token distribution is adjusted by the pointwise difference between the with-context and no-context distributions, scaled by the Additive CAD parameter $\gamma \in \mathbb{R}$.

2.1.3 Decoding by Contrasting Layers (DoLa)

Decoding by Contrasting Layers (DoLa) is a method introduced by Chuang et al. (2023), shown to improve truthfulness across a variety of multiple-choice and open-ended generation tasks. The approach contrasts the final output distribution of a language model with the output distribution produced by an earlier *transformer layer*, using a pointwise-ratio technique similar to CAD (see Section 2.1.1).

Transformer-based language models consist of multiple *layers*, each applying transformations to the vector representations of input tokens. An *early exit* at layer l involves decoding the output at that layer into a probability distribution over tokens, denoted $\mathrm{P}^{(l)}(y_t)$. The number of layers is fixed at design time—e.g., LLaMA-2-7B includes 32 layers (Touvron et al., 2023). DoLa computes a new distribution by contrasting this early-layer distribution with the final output distribution $\mathrm{P}(y_t)$ as follows:

$$\tilde{\mathrm{P}}_{\mathrm{DoLa}}(y_t) \propto \mathrm{P}(y_t)\left(\frac{\mathrm{P}(y_t)}{\mathrm{P}^{(l)}(y_t)}\right)$$

The layer l is dynamically selected at each decoding step as the one whose output distribution maximizes the *Jensen–Shannon divergence* (JSD) from the final output. The JSD is a symmetric variant of the Kullback–Leibler (KL) divergence $D(P \mid\mid Q)$ between two distributions P and Q, defined as:

$$\mathrm{JSD}(P \mid\mid Q) = \frac{1}{2}D(P \mid\mid M)+\frac{1}{2}D(Q \mid\mid M)$$

where $M = \frac{1}{2}(P + Q)$ is the mixture distribution.

Hugging Face's `transformers` library offers a convenient way to implement DoLa in practice [1]. The user may optionally restrict the premature layer selection to either the lower or upper half of the model's layers. For all DoLa results reported in this paper, layer selection was limited to the lower half for conciseness and consistency.

[1]See `https://huggingface.co/docs/transformers/en/generation_strategies#dola` for implementation details.

2.1.4 Adaptive Plausibility Constraint (APC)

Li et al. (2022) proposed a method to prevent otherwise unlikely tokens from being amplified by the contrastive decoding process.

Formally, given a probability distribution after some form of contrastive decoding is applied $\tilde{\mathrm{P}}(y_t)$ over the set of possible next tokens $y_t \in \mathcal{V}$, the APC sets token probabilities to zero if the original token probability given the context doesn't exceed some threshold $\beta \in [0, 1]$:

$$\tilde{\mathrm{P}}_{\mathrm{APC}}(y_t) \propto \begin{cases} \tilde{\mathrm{P}}(y_t) & \text{if } \mathrm{P}(y_t \mid \mathbf{c}, \mathbf{x}, \mathbf{y}_{<t}) > \beta \\ 0 & \text{otherwise} \end{cases}$$

Throughout this paper whenever CAD or Additive CAD is applied, we set the APC threshold to $\beta = 0.1$ in accordance with prior work on contrastive decoding (Chuang et al., 2023; Li et al., 2022).

2.2 Experimental Setup

2.2.1 Datasets

The CAD and Additive CAD methods are tested on datasets that specifically provide both an input prompt $\mathbf{x}$ and some form of context $\mathbf{c}$ which then allow for the LLM output distributions with and without context to be contrasted against one another.

Knowledge conflicts We evaluate performance on the MemoTrap dataset (Liu and Liu, 2023). MemoTrap is designed to test the susceptibility of LLMs to fall into memorisation traps. For instance, a typical question is "Write a quote that ends in the word "thoughts": Actions speak louder than". It is common for an LLM's output to be "words" (which is the standard ending of the classic proverb "Actions speak louder than words"), despite the prompt specifically instructing that a different ending be made. The tendency for models to defy instructions as in this example is viewed as undesirable behaviour.

This dataset has the unusual feature that model performance on the dataset scales inversely to model size.

Table 1: Example prompts $\mathbf{x}$, contexts $\mathbf{c}$ and correct (gold) answers $\mathbf{g}$ for the MemoTrap and Natural Questions datasets. The prompts and context form the inputs to CAD and Additive CAD according to the implementation details provided in section 2.1.

MemoTrap	
$\mathbf{c}$	Write a quote that ends in the word "early":
$\mathbf{x}$	Better late than
$\mathbf{g}$	early
Natural Questions	
$\mathbf{c}$	“ Theme from Mission : Impossible ” is the theme tune of the TV series Mission : Impossible (1966 – 1973) . The theme was written and composed by Argentine composer Lalo Schifrin and has since gone on to appear in several other works of the Mission : Impossible franchise , including the 1988 TV series , the film series and the video game series . The 1960s version has since been acknowledged as one of TV ’s greatest theme tunes .
$\mathbf{x}$	who wrote the theme song for mission impossible?
$\mathbf{g}$	Lalo Schifrin

Question-answer The Natural Questions dataset (Kwiatkowski et al., 2019) is a QA dataset. The context in this case is an excerpt of a Wikipedia article on the topic to aid in the answering of a particular general knowledge question. This dataset therefore tests on a model's ability to answer general knowledge questions given additional information, which in practice could be provided by—for instance—a basic RAG system which pulls Wikipedia article excerpts based on the cosine similarity of encoded representations of the Wikipedia excerpt to the input prompt.

Example prompts, contexts and correct answers for the MemoTrap and Natural Questions dataset are provided in table 1.

2.2.2 Metrics

The two main metrics used in the results of section 2.3 are an *exact match* (EM) score, and a more relaxed *substring match* score. Both methods are described in detail in this section. Both metrics ignore character case, punctuation, articles and extra whitespace.

Exact Match (EM) score We adopt a multi-reference exact match score, following the widely-used definition introduced by Rajpurkar et al. (2016) and its implementation in the official evaluation script[2]. A score of 1 is assigned if the model's output exactly matches *any* of the gold reference answers, and 0 otherwise. This metric is chosen as it matches the metric employed by the original CAD paper (Shi et al., 2024) on the same MemoTrap and Natural Questions benchmarks.

Formally, let $G^{(q)} = \{g_1^{(q)}, g_2^{(q)}, \ldots, g_n^{(q)}\}$ be the set of *normalised*[3] gold answer strings for question q of a particular dataset, and let $a^{(q)}$ be the normalised string of the model's decoded predicted answer. For a Q-question dataset, we define a multi-reference exact-match score for question q as follows:

$$\mathrm{Match}^{(q)} = \begin{cases} 1 & \text{if } a^{(q)} \in G^{(q)} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

And the overall EM score across the dataset:

$$\mathrm{AvgMatch} = \frac{1}{Q}\sum_{q=1}^{Q} \mathrm{Match}^{(q)} \tag{2}$$

If there is only *one* gold answer $g^{(q)}$ for each question in the dataset, the multi-reference exact match score collapses into the single-reference EM score for one gold answer, with the EM match for each question computed as:

$$\mathrm{Match}^{(q)} = \begin{cases} 1 & \text{if } a^{(q)} = g^{(q)} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

and the overall accuracy similarly computed as the mean of these match scores across all Q questions in the dataset.

Since there are multiple gold answers for

[2] See https://github.com/rajpurkar/SQuAD-explorer/blob/master/evaluate-v2.0.py#L98 for the exact implementation.

[3] Lowercased string with punctuation, articles and extra whitespace removed.

some questions in the Natural Questions dataset, the EM score for Natural Questions is computed as in 1.

However, since there is only one gold answer for each question in the MemoTrap dataset, the simplification in equation 3 is applied.

Substring match score The substring match score is a more relaxed score which allows for other characters in an LLM answer to surround an otherwise correct substring. For example, if the normalised gold answer string $g^{(q)}$ to question q in the dataset is `'1971'`, then this metric would mark the normalised LLM answer $a^{(q)}$ of `'year 1971'` as correct.

Formally, let $G^{(q)} = \{g_1^{(q)}, g_2^{(q)}, \ldots, g_n^{(q)}\}$ be the set of normalised gold answer strings for question q of a particular dataset, and let $a^{(q)}$ be the normalised string of the model's decoded predicted answer. For a Q-question dataset, we define the substring match score as follows:

$$\text{Match}^{(q)} = \begin{cases} 1 & \text{if any } g_i^{(q)} \text{ is a substring of } a^{(q)} \\ 0 & \text{otherwise} \end{cases}$$

And the overall accuracy score is computed as the mean of these scores across the entire question dataset, as in equation 2.

2.2.3 Models

The primary language model used in this study is Meta's LLaMA-2-7B (Touvron et al., 2023), selected for its public availability and relatively low computational inference cost. LLaMA-2-13B (Touvron et al., 2023) is employed in Section 2.3.6 to align with the implementations of the studies discussed therein.

2.2.4 Other Experimental Details

Temperature We set the temperature of both LLM output distributions to 1 throughout the experiments on CAD, as in the original CAD paper (Shi et al., 2024). We leave the study of the effects of varying the temperature of the LLM output distributions with or without context to future work.

Prompts For the MemoTrap benchmark, we require no additional prompting other than simple concatenation of the context $\mathbf{c}$ and question $\mathbf{x}$ for the LLM output distribution with context, and simply the question $\mathbf{x}$ alone for the output distribution without context. Hence the prompt templates used are as provided in Table 5 (appendix A.1) and match those of the original CAD paper (Shi et al., 2024).

For the Natural Questions benchmark, however, we wish to simulate a realistic RAG system with clear separation of the contextual information from the question itself. Hence the prompt template for the LLM output distribution with context clearly instructs the LLM to first read the context and then answer the corresponding question, clearly separated by a new line and "Q:" to signify the end of the context and the beginning of the question. The prompt for the LLM output distribution without context is designed to match the wording closely but without reference to any additional context. The full template is provided in Table 8 (appendix A.1) and is the prompt used for Natural Questions unless stated otherwise. Section 2.3.6 presents an evaluation of how prompt style affects performance of CAD on the Natural Questions benchmark.

Stopping Criterion In order to prevent the CAD output from continuing indefinitely and to save computing costs, well-defined stopping criterion on the generation are required. Unless stated otherwise, the maximum number of generated tokens per question is 20. Additionally, we define stopping symbols in natural sentence breaks so that the CAD response doesn't include any ancillary explanation or follow-up which would prevent an exact match with the gold answer despite a correct answer being initially given. Hence, generation on the MemoTrap benchmark stops on a `'.'` token, and generation on the Natural Questions benchmark stops once a `'.'` or a newline appears in the decoded output string.

Handling of infinite float representations Since the logit values of an LLM output may occasionally be a float representation of positive or negative infinity, errors may arise when attempting to add or subtract finite-valued logit values from these non-finite logits. The solution employed in this paper is to preemptively set positive infinite floats to 1,000 and negative infinite floats to $-1{,}000$ before performing any floating-point arithmetic on the logits. We found that this setting was sufficient in magnitude to avoid any potential unwanted side-effects of the truncation.

2.3 Results and Discussion

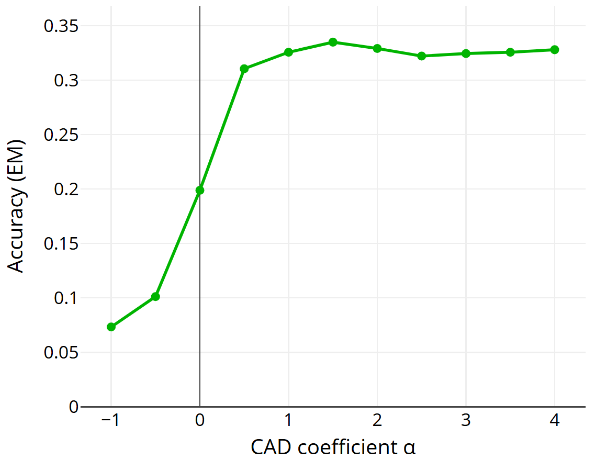

Figure 1: Accuracy of LLaMA-2-7B on the Memotrap benchmark at temperature 0.

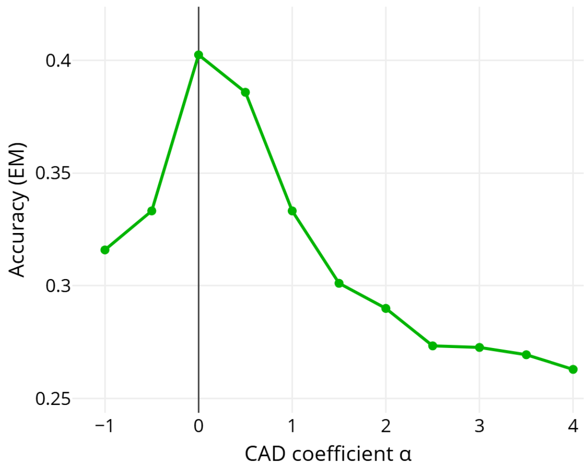

Figure 2: Accuracy of LLaMA-2-7B on the Natural Questions benchmark at temperature 0.

2.3.1 Reproduction of CAD Results on Memotrap Benchmark

The effect of varying the CAD coefficient on the Memotrap benchmark accuracy is shown in Figure 1.

2.3.2 Reproduction of CAD Results on Natural Questions Benchmark

The effect of varying the CAD coefficient on the Natural Questions benchmark accuracy is shown in Figure 2.

Inconsistencies across papers

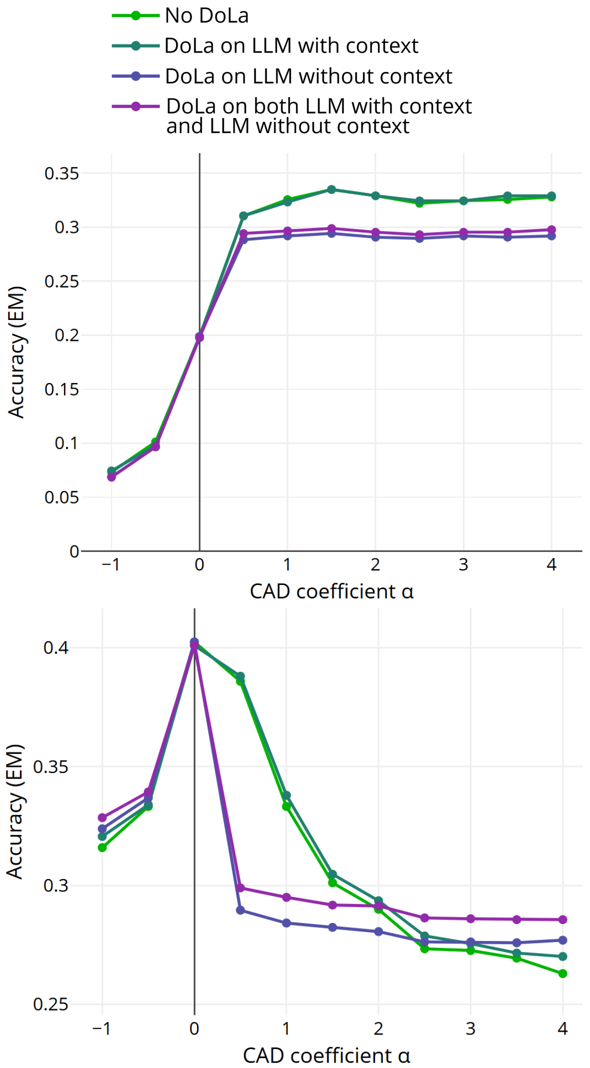

Figure 3: Accuracy of LLaMA-2-7B on the Memotrap benchmark (above) and Natural Questions benchmark (below) at temperature 0, for various CAD-DoLa schemes.

2.3.3 Novel Method: CAD-DoLa

The effect of varying the CAD coefficient for various DoLa schemes on Memotrap and Natural Questions benchmark accuracy is shown in Figure 3.

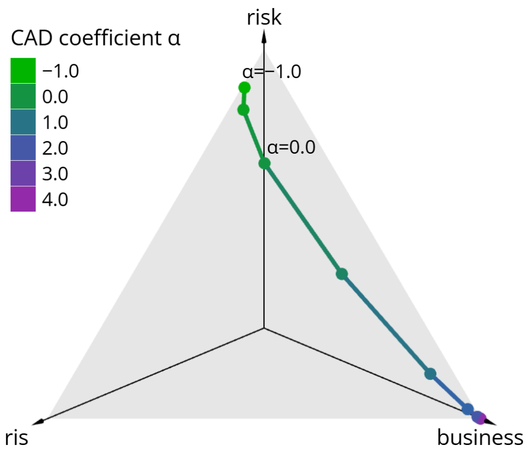

Figure 4: Probability distribution of top three output tokens of LLaMA-2-7B with CAD applied.
Context: *Write a quote that ends in the word "business":*
Prompt: *Advisers run no*

2.3.4 Probability Simplex Visualisation

The probability simplex is a convenient way to visualise a probability distribution.

Successful CAD An example of a successful journey through the probability simplex as a result of CAD is shown in Figure 4.

Partially successful CAD An example of a partially-successful journey through the probability simplex as a result of CAD is shown in Figure 5 (above).

Unsuccessful CAD-DoLa An example of an unsuccessful journey through the probability simplex as a result of CAD-DoLa is shown in Figure 5 (below). The question would have been answered correctly for $\alpha = 1.0$ had no DoLa been applied, as seen in the above half of the figure. Here, DoLa is applied to the high layers of the transformers on both the LLMs with and without context. This shifts the two primary output distributions ($\alpha = -1.0$, $\alpha = 0.0$) to new locations.

Motivation for Additive CAD A particularly interesting journey through the probability simplex is shown in Figure 6.

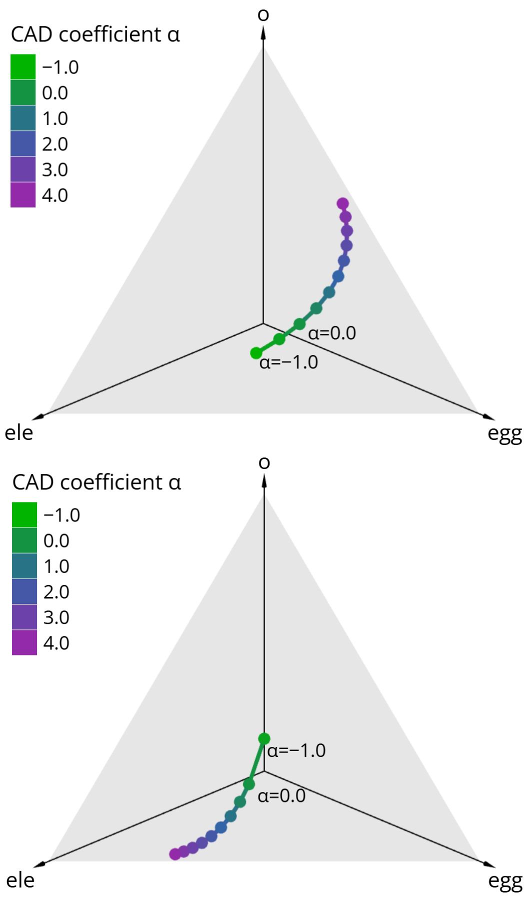

Figure 5: Probability distributions of top three output tokens of LLaMA-2-7B with CAD applied. Above: no DoLa. Below: DoLa on both LLM with context and LLM without context.
Context: *Write a quote that ends in the word "egg":*
Prompt: *He that steals an egg will steal an*

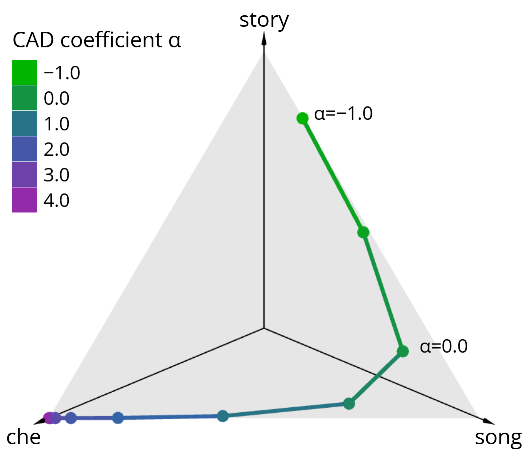

Figure 6: Probability distribution of top three output tokens of LLaMA-2-7B with CAD applied. DoLa on both LLM with context and LLM without context.
Context: *Write a quote that ends in the word "song":*
Prompt: *Applause is the echo of a*

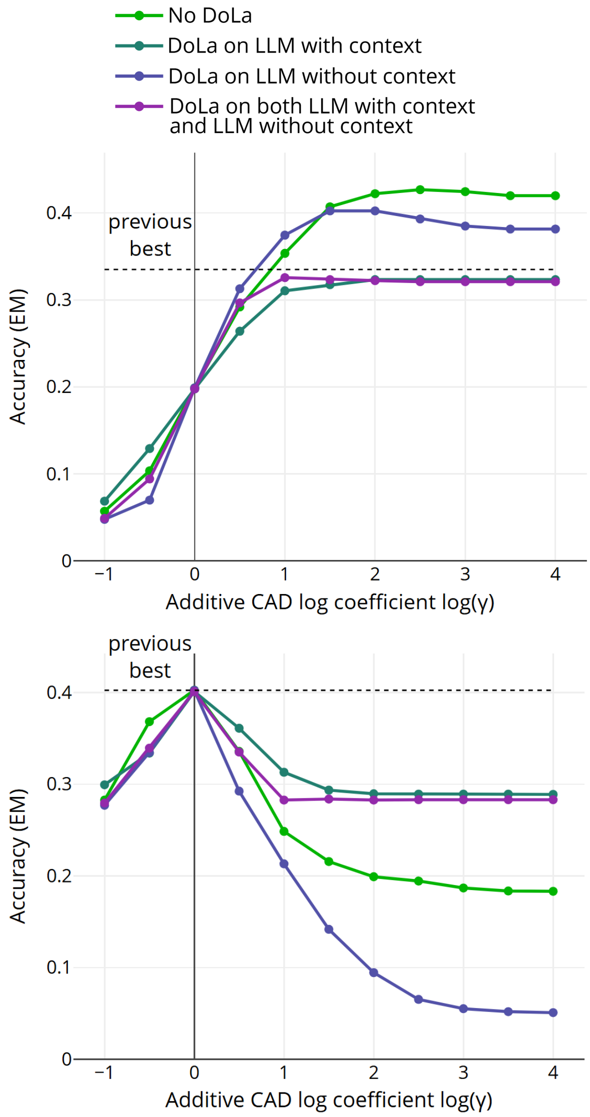

Figure 7: Accuracy of LLaMA-2-7B on the Memotrap benchmark (above) and Natural Questions benchmark (below) at temperature 0, for various Additive CAD-DoLa schemes.

2.3.5 Novel Method: Additive CAD

The effect of varying the Additive CAD coefficient for various DoLa schemes on Memotrap and Natural Questions benchmark accuracy is shown in Figure 7.

Additive CAD-DoLa on Memotrap Benchmark

Additive CAD-DoLa on Natural Questions Benchmark

Table 2: Accuracy (EM) of Llama-2-13B on the Natural Questions benchmark, across different publications. Note that decoding and evaluation settings vary across each paper.

Paper	Regular ($\alpha = 0$)	CAD ($\alpha = 1$)
CAD (Shi et al., 2024)	0.223	0.336
COIECD (Yuan et al., 2024)	0.731	0.675
AdaCAD (Wang et al., 2024)	0.443	0.379

Table 3: Accuracy (substring match) of Llama-2-13B on the Natural Questions benchmark, by input prompt style (each of which is provided in appendix A.1), with all other decoding and evaluation settings kept constant.

Input prompt style (paper)	Regular ($\alpha = 0$)	CAD ($\alpha = 1$)
CAD (Shi et al., 2024)	0.341	0.424
COIECD (Yuan et al., 2024)	0.780	0.607
AdaCAD (Wang et al., 2024)	0.785	0.709
This paper	0.771	0.699

2.3.6 Addressing Inconsistencies Across Papers

3 Wider Exploration of Idea Space with Doubt Injection

3.1 Method

This section introduces *Doubt Injection*, a method for modifying the generative trajectory of LLMs by inserting uncertainty cues into the output text.

The approach operates during standard token-by-token generation, but at natural discourse boundaries (e.g., paragraph breaks), it probabilistically appends phrases such as "But " to subtly redirect the model's reasoning or introduce hesitation. This mechanism is designed to steer the model toward more reflective continuations without altering the original prompt.

An implementation of this method is provided in Algorithm 1.

Algorithm 1 Doubt Injection During LLM Text Generation

Require: Prompt $\mathbf{x}$, maximum generation steps N, injection probability p
Ensure: Generated sequence $\mathbf{y}$
  Initialize $\mathbf{y} \leftarrow [\ ]$
  Let $\mathcal{S} \leftarrow \{$"`.\n\n`", "`. \n\n`", "`\n\n`", "` \n\n`"$\}$
  for $t = 1$ to N :
    Sample $y_t \sim \mathrm{P}(y_t \mid \mathbf{x}, \mathbf{y})$
    Append y_t to $\mathbf{y}$
    Let $s \leftarrow$ `decode`(y_t)
    if $s \in \mathcal{S}$:
      Sample $u \sim \mathcal{U}(0, 1)$
      if $u < p$:
        Append `tokenize`("`But `") to $\mathbf{y}$
    if y_t is the end-of-sequence token :
      break
  return $\mathbf{y}$

3.2 Experimental Setup

3.2.1 Datasets

Reasoning and social intelligence SimpleBench (Philip and Hemang, 2024) is a multiple-choice benchmark test (6 choices per question) on which humans with high school knowledge consistently outperform state-of-the-art LLMs. The questions are designed to cover 'spatio-temporal reasoning', 'social intelligence', and 'linguistic adversarial robustness (or trick questions)'.

However, only a small portion (10 questions) are made available to the public. A small portion of these questions were found to never produce a correct answer and hence in the interest of saving compute, the subset of questions whose results are shown (referred to herein as 'the subset of SimpleBench questions') are questions 1, 2, 3, 8, 9 and 10 of the publicly available questions.

The prompt template used on this benchmark is provided in the appendix in table 10. The LLM is instructed to contain its reasoning within <think> and </think> tags and its final answer within <answer> and </answer> tags. In evaluating the responses, the string within the final answer tags is extracted, stripped of any leading and trailing whitespace and compared to the correct answer which is one of A, B, C, D, E, or F.

Mathematical reasoning The American Invitational Mathematics Examination (AIME) is a prestigious, invite-only mathematics competition for high-performing high-school students. The AIME 2024 benchmark (MAA, 2024) contains 30 challenging mathematics problems, which each typically require around 10 steps each to solve and have an answer which is an integer in the range 0 to 999.

This dataset holds the major advantage over other popular mathematics benchmarks in that performance on it has not yet become saturated. For instance, given that most new models now readily achieve over 90% accuracy on MATH-500, differences in model performance are difficult to ascertain. Due to the increased difficulty and multi-step reasoning process required to solve the problems in the AIME benchmarks however, there is still much improvement yet to be made, and there is great variation in performance across state-of-the-art models.

The prompt used on an example question is provided in the appendix in table 11. The LLM is instructed to reason step-by-step put the final answer within `\boxed{}` from which the answer is then extracted and stripped of any leading and trailing whitespace. This string is then converted to an integer with python's `int` function (the answer is immediately deemed incorrect the function returns an error), in order to account for LLM answers that are given in a different form to the correct answers given in the dataset but should nevertheless be marked as correct. For instance `73.0` and `073`.

Due to the low number of questions in each benchmark, multiple runs are made through each datasets, with the final accuracy computed as specified in section 3.2.2. This will therefore increase the precision of our results and reduce the size of the error bars (calculated through the method outlined in section 3.2.5).

3.2.2 Metrics

Accuracy Calculation To account for the variance in model performance due to stochastic decoding, we report the average pass@1 score across multiple runs. Specifically, for each run, the model is evaluated on the full benchmark dataset by generating a single response per question, and the pass@1 metric is computed as the proportion of questions for which the model's first response is correct. Given Q questions in the dataset and letting $c_q \in \{0, 1\}$ indicate whether the answer to question q is correct (1 if correct, 0 otherwise), the pass@1 score for a single run is calculated as:

$$\text{Pass@1} = \frac{1}{Q}\sum_{q=1}^{Q} c_q$$

The average pass@1 is then defined as the mean of these pass@1 scores across all n independent runs:

$$\text{AvgPass@1} = \frac{1}{n}\sum_{i=1}^{n} \text{Pass@1}_i$$

This is mathematically equivalent to the overall proportion of correct answers across all runs combined.

3.2.3 Models

The experiments conducted on Doubt Injection are required to be carried out with models that allow for token-by-token control. This would be highly inefficient with the use of closed-source models via API (due to network latency and lack of caching with single-token requests). Moreover, many closed-source reasoning model providers such as OpenAI's o1 do not provide access to the full chain-of-thought, where Doubt Injection is intended to operate.

The models chosen are two distilled versions of DeepSeek-R1 (Guo et al., 2025): DeepSeek-R1- Distill-Qwen-32B and DeepSeek-R1- Distill-Qwen-1.5B. These models satisfy the requirement of being open source models that produce chains-of-thought. The distilled models are selected over the original model because of their lower inference costs as a result of fewer model parameters. In addition, the distilled models perform strongly on benchmarks compared to alternative models of similar size.

3.2.4 Nucleus Sampling (top-p) Setting

In alignment with the evaluation setup of the original DeepSeek-R1 paper (Guo et al., 2025), we set the top-p (nucleus sampling) value to 0.95 in our experiments. Introduced by Holtzman et al. (2019), nucleus sampling selects tokens from the smallest possible subset of the vocabulary whose cumulative probability exceeds a threshold p. Given a model's predictive distribution $\mathrm{P}(w_t \mid \mathbf{w}_{<t})$ over the vocabulary $\mathcal{V}$, top-p sampling defines $\mathcal{V}^{(p)} \subset \mathcal{V}$ as the smallest set such that:

$$\sum_{w \in \mathcal{V}^{(p)}} \mathrm{P}(w_t = w \mid \mathbf{w}_{<t}) \geq p$$

and samples the next token from $\mathcal{V}^{(p)}$. This ensures that the sampled tokens remain sufficiently relevant.

3.2.5 Calculation of Confidence Intervals

When evaluating the accuracy of LLMs on question-answer datasets, many papers do not display error bars on graphs. This can be highly misleading. Consider comparing the accuracies of two different LLMs (at a non-zero temperature) on a 200-question dataset. If the resulting accuracies were found to be 72.5% and 73.0% for LLM A and LLM B respectively, one may conclude that LLM B is more accurate than LLM A on this dataset. However, after evaluating the extent of the error in this claim, we discover that there is only a 54% chance that LLM B is in fact more accurate than LLM A on this dataset, comparable to the uncertainty in the flipping of a coin. Most major papers and popular commentators fail to acknowledge this and will often take minor improvements on a benchmark as proof of one LLM's superiority over another, regardless of the number of experiments run.

This result transpires if we consider the probability distributions in LLM accuracy given a set of observations, as explained below:

It is assumed that there exists some true accuracy $a \in [0, 1]$ of an LLM for a given question-answer dataset. This is the probability that the LLM answer on a uniform random sampled question will be correct.

If we now make a set of observations $\mathcal{D}$, of N correct answers and M incorrect answers on this dataset, then the probability of making this set of observations is:

$$\mathrm{P}(\mathcal{D}|a) = a^N (1-a)^M$$

Using Bayes' theorem, the posterior probability distribution of a given our observations is:

$$p(a|\mathcal{D}) \propto \mathrm{P}(\mathcal{D}|a)p(a)$$

If we assume a uniform prior, (no initial bias about what a might be), then:

$$p(a|\mathcal{D}) \propto a^N (1-a)^M$$

This posterior distribution follows the Beta distribution with parameters $\alpha = N + 1$ and $\beta = M + 1$:

$$p(a|\mathcal{D}) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} a^{\alpha-1} (1-a)^{\beta-1}$$

Where Γ is the gamma function. The peak of this distribution occurs at value $\hat{a}$, found by maximising the posterior with respect to a:

$$\hat{a} = \frac{N}{N + M}$$

That is, the proportion of correct answers. This is our maximum-likelihood estimate of the true accuracy and matches the average pass@1 accuracy described in section 3.2.2.

Using the cumulative density function of the Beta distribution to find at what values of a the cumulative probability is at 2.5% and 97.5%, we obtain the lower and upper bounds of the 95% confidence interval respectively. This is the method used in this paper to calculate confidence intervals for LLM accuracy.

We can also (via Monte Carlo simulation or other methods), use two Beta distributions to estimate the probability that one LLM accuracy is higher than another. This transcribes to the probability that some method improves the accuracy on a particular dataset.

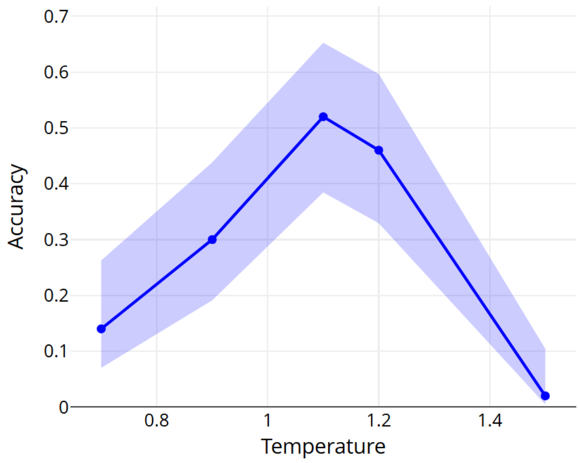

Figure 8: Accuracy of DeepSeek-R1-Distill-Qwen-32B on the altered river crossing puzzle, for different temperatures. 50 runs for each temperature.

3.3 Results and Discussion

3.3.1 Motivating Results (the Altered River Crossing Problem)

In order to motivate Doubt Injection as a potentially desirable method, we carry out a brief case study on a particular prompt which is designed to break with the LLM's training data and test the model's ability to reason on an unusual and unseen problem. The question and example responses deemed correct and incorrect are provided in table 4. The question is based on a classic river-crossing problem, where the solution to the classic problem involves a seven-step process to transport various items back and forth across a river, to prevent adverse interactions between some of the items and safely get all items to the other side. In this altered problem, however, the style of the wording is kept similar but is made trivially-solvable in a single step.

The full input prompt is written to elicit chain-of-thought with convenient formatting, and is based on the prompts used in the training of Deepseek-R1-Zero (Guo et al., 2025). It is supplied in the appendix in table 9.

50 responses at various temperatures are manually labeled (due to the variation in wording of possible correct responses). The results are provided in Figure 8.

The peak accuracy is 52% at a temperature of around $T = 1.1$. The majority of incorrect

Table 4: Prompt and typical answers (the portion of the LLM response enclosed within `<answer>` and `</answer>` tags) of DeepSeek-R1-Distill-Qwen-32B for the altered river crossing problem.

Prompt: Solve the following puzzle: A man and a goat are on the left side of a river. There is a wolf and a cabbage on the right side of the river. The man has a boat. The boat can carry only the farmer and a single item. How can the farmer get the goat to the right side of the river?
Example correct answer: The farmer takes the goat across the river to the right side.
Example incorrect answer: 1. The farmer takes the goat across the river to the right side. 2. The farmer returns alone to the left side. 3. The farmer takes the wolf across to the right side. 4. The farmer brings the goat back to the left side. 5. The farmer takes the cabbage to the right side. 6. The farmer returns alone to the left side. 7. The farmer takes the goat to the right side. All three—the goat, wolf, and cabbage—are safely across the river.

answers at this temperature conclude with a 7-step solution which is typical of the classical problem but not correct in this altered version. This suggests that the LLM's training data is strongly influencing its response—the LLM likely having been trained on a number of descriptions of the classic river problem yet none of the altered problem. A small proportion of the incorrect answers (around 17%) falsely claim that the problem is impossible to solve.

At temperature $T = 0.7$, far more answers conclude with the incorrect 7-step solution. At the higher temperature of $T = 1.5$, however, the responses are mostly nonsensical with large amounts of language-mixing (a phenomenon acknowledged in the original DeepSeek-R1 paper (Guo et al., 2025)).

Exploration of Idea Space An evaluator LLM (Qwen2.5-32B at a zero temperature setting) was used to analyse the chain-of-thoughts of the altered river crossing problem. After splitting up the chain-of-thought into overlapping segments, we query the evaluator LLM on each segment as to whether a particular idea is mentioned in the text.

This results in a true/false for each idea, for each response. This is then averaged over the 50 responses, before being normalised to form a 'distribution over ideas' at each temperature, shown in Figure 9.

We can see from this figure that the vast majority of ideas explored in the chain-of-thought at lower temperatures are concentrated in a small few. However, as the temperature rises, the distribution flattens, and the variation in ideas explored becomes richer.

This result is expected from the fact that higher temperature results in higher randomness in individual tokens. If a wider variety of tokens are explored, it follows that a wider variety of 'ideas' would result in the output as a whole.

Relation to accuracy The individual chosen ideas were chosen to include ideas from both correct and incorrect responses. The idea "the setup in the classic version being different" (an important realisation in correctly solving the puzzle) shows increases in prevalence as the temperature rises. This coincides with the increase in accuracy seen over this temperature range in Figure 8. This suggests that a higher temperature allows this and other under-explored correct ideas to be accessed more often by the chain-of-thought, resulting in a higher accuracy.

Statements of doubt The mention of "I'm confused" also increases in prevalence as the temperature rises. This provides a motivation for Doubt Injection. If the chain-of-thought can be made to explore a *richer set of ideas*, perhaps it will be more likely to mention the correct idea, explore it further, and ultimately conclude with it. The following sections test

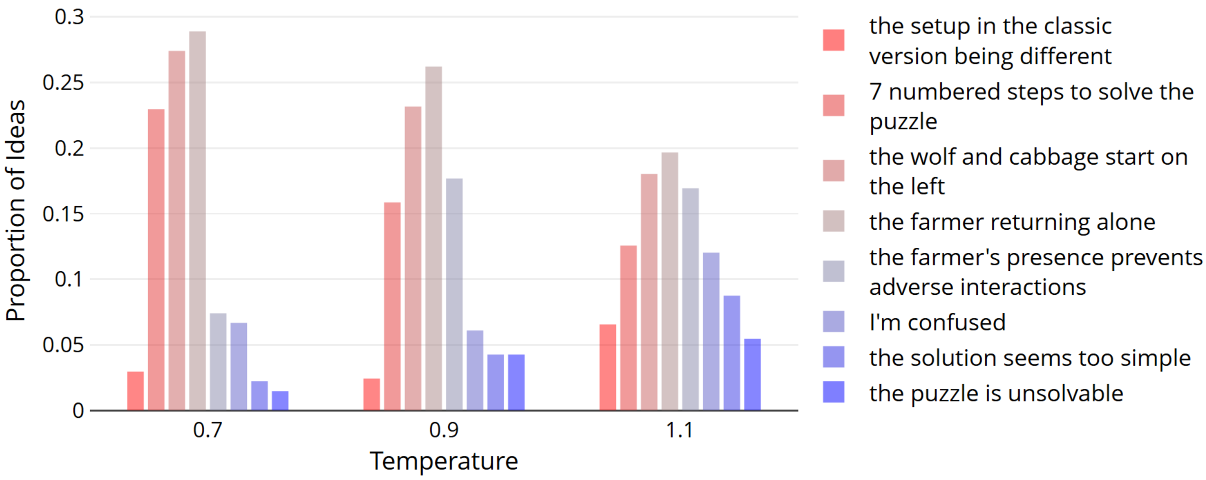

Figure 9: The proportional makeup of 'ideas' explored in the chain-of-thought of DeepSeek-R1-Distill-Qwen-32B in answering the altered river crossing puzzle, across different temperatures.

whether the injection of a statement encouraging a sense of doubt, for instance "I'm confused." can cause a wider variety of ideas to be explored, and ultimately increase the accuracy of LLMs.

3.3.2 Doubt Injection: Effect of Temperature

We begin by comparing a temperature-accuracy response of no doubt injection, and an injection of "But wait, let me think again." with probability $p = 0.5$ at each new paragraph. Results on the subset of SimpleBench questions is shown in Figure 10.

A temperature of 1.0 has highest performance on this dataset. Major degradation in performance is observed at temperatures higher than 1.2. High temperatures are far more likely to continue generating tokens passed the 10,000 limit set in this experiment.

Consistently across temperatures, Doubt Injection of this particular string at a probability of 0.5 worsens performance on this question set.

3.3.3 Effect of Injection String

The effect of Doubt Injection string on the accuracy in answering the subset of SimpleBench questions is shown in Figure 11 (above). The injection of "But" achieves the highest accuracy. However, this improvement is marginal, with an accuracy of 26.7% over 26.1% with no doubt injection. Comparing probability distributions in accuracy, we can say with only 58.1% certainty that Doubt Injection results in higher accuracy for these questions.

However, some specific questions show a clear advantage for Doubt Injection. The same graph, but for question 9 only, is shown in Figure 11 (below). For this question, an accuracy of 16.7% can be increased to 28.0% by randomly injecting the string "But" into the chain-of-thought. Comparing probability distributions, we can say with 95.4% certainty that Doubt Injection results in higher accuracy for this question.

For 3 out of the 6 questions, the injection of "But" makes an improvement to the accuracy, and for 1 out of the 6, there is no difference. Question 8 is the only question for which none of the tested Doubt Injection strings make an improvement over no injection. In general, the performance of doubt injection—and the best-performing injection string—is highly dependent on the specific question asked.

3.3.4 Effect of Injection Probability

The accuracy of injection string "But" is now tested for different doubt injection probabilities. The result is shown in Figure 12 (above). We see an initial worsening of performance for low doubt injection probability, followed by a gentle peak at $p \simeq 0.3$ and a degradation in performance by $p = 0.5$.

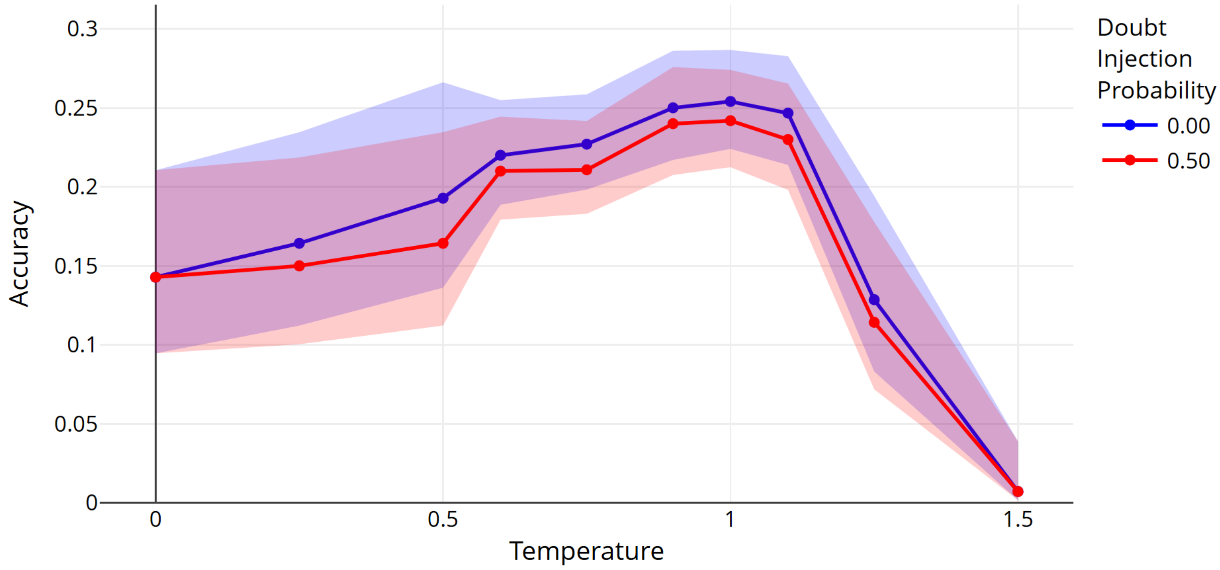

Figure 10: Accuracy of DeepSeek-R1-Distill-Qwen-32B on the subset of SimpleBench questions, with and without Doubt Injection of "But wait, let me think again." Varying confidence interval widths are as a result of varying number of runs at each temperature.

Again, the increase in accuracy on the subset of SimpleBench questions is not significant when doubt injection is incorporated. However, the performance of question 9 only, shown in Figure 12 (below), shows a clear performance gain when a doubt injection probability in the range 0.2 to 0.4 is used.

This again shows that although certain applications can greatly benefit from Doubt Injection in the chain-of-thought, the method may have a poor ability to generalise across wider datasets.

3.3.5 AIME 2024 Benchmark Performance

The AIME 2024 dataset contains 30 complex mathematical problems. The effect of varying the doubt injection probability of "But" into the chain-of-thought is shown in Figure 13.

Although for the 32B model there is a marginal increase in accuracy with an injection probability of $p = 0.05$, this is statistically very insignificant. The general trend for both the 32B and 1.5B models is that accuracy decreases as injection probability increases. The lower proportional decrease in accuracy of the 32B model, compared with the 1.5B model, suggests it is more robust to alterations in its chain-of-thought.

Interestingly, however, the trend is again highly question-dependent. Some questions show a clear downward trend in accuracy, whilst others show a clear upward trend. For some questions with the 32B model, the performance peaks at $p = 0.05$. Notably for problem 11, doubt injection of $p = 0.05$ increases the accuracy from 0% to 50% compared with no doubt injection ($p = 0$). Statistically, we have 91.7% certainty that doubt injection improves the accuracy for this particular problem.

However, the exact nature or topics of the questions with upward or downward trends show no clear pattern. Doubt injection shows a poor ability to generalise across the whole dataset. As with the SimpleBench questions, therefore, there is no clear evidence that doubt injection of this selected injection string increases the overall performance of the model.

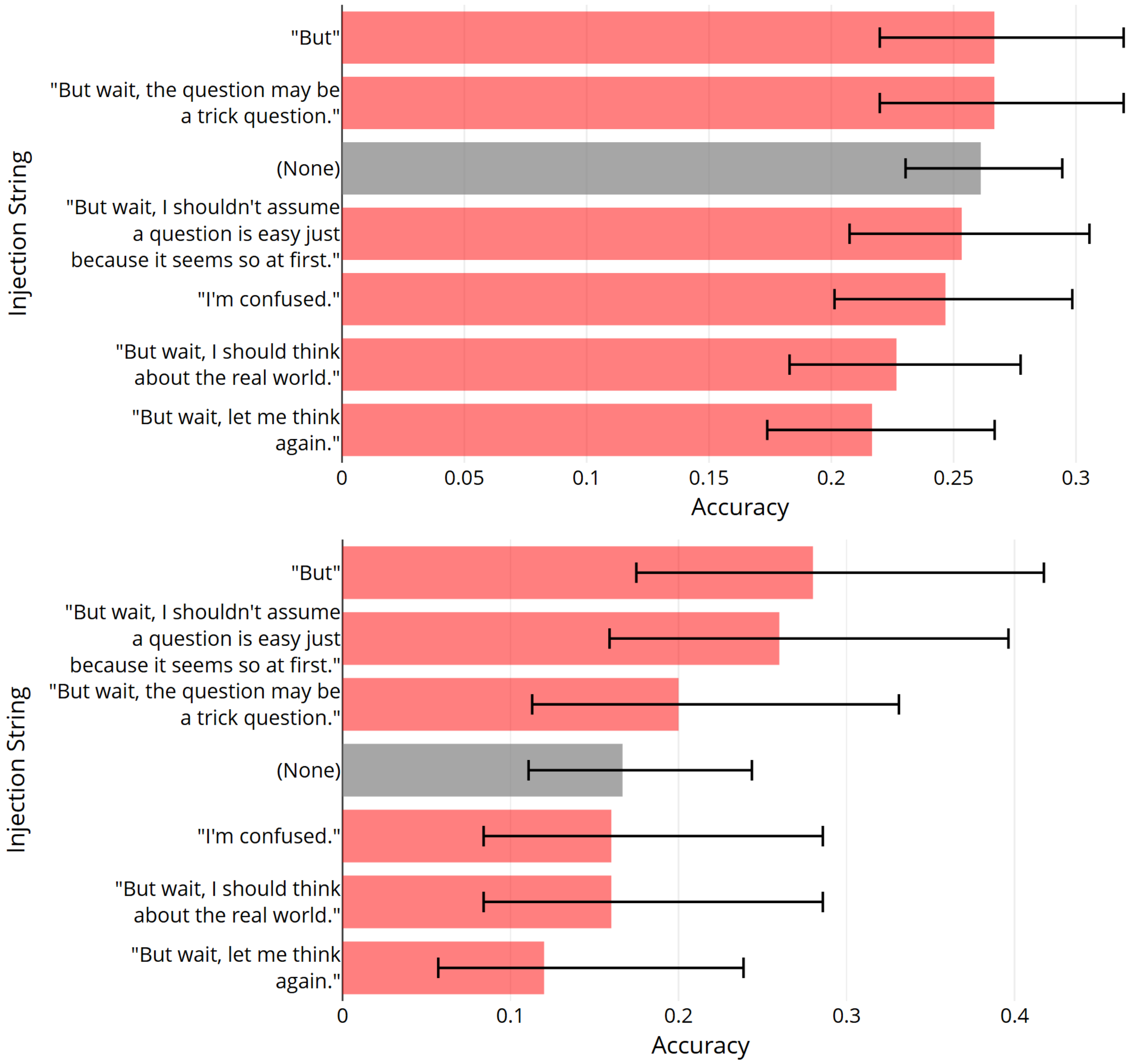

Figure 11: Accuracy at a temperature of 1.0 of DeepSeek-R1-Distill-Qwen-32B on the subset of SimpleBench questions (above), and question 9 only (below), for different doubt injection strings. 50 runs were made for injections with $p = 0.25$ (red), and 120 runs for no injection string (grey).

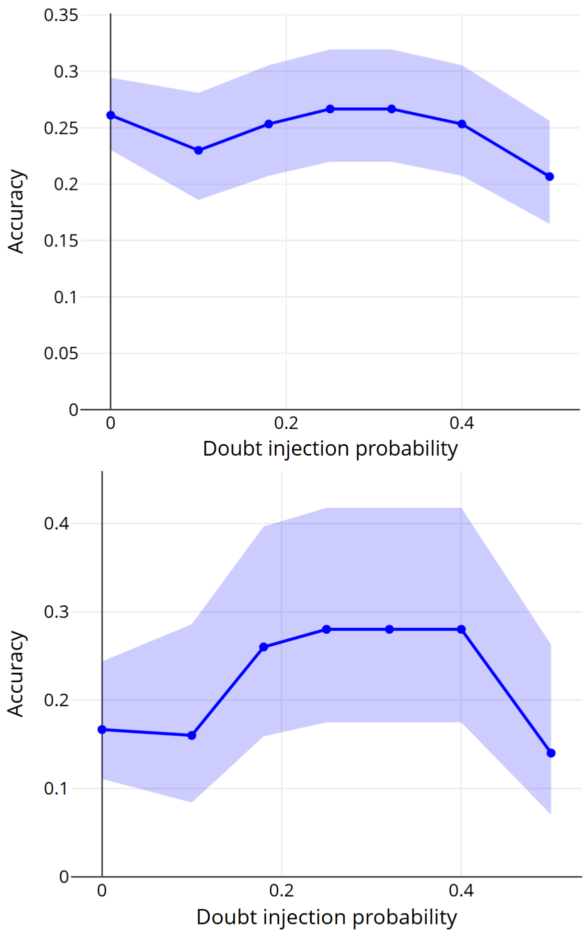

Figure 12: Accuracy of doubt injection string "But" on the subset of SimpleBench questions (above) and question 9 only (below) against probability of injection. DeepSeek-R1-Distill-Qwen-32B, temperature 1.0.

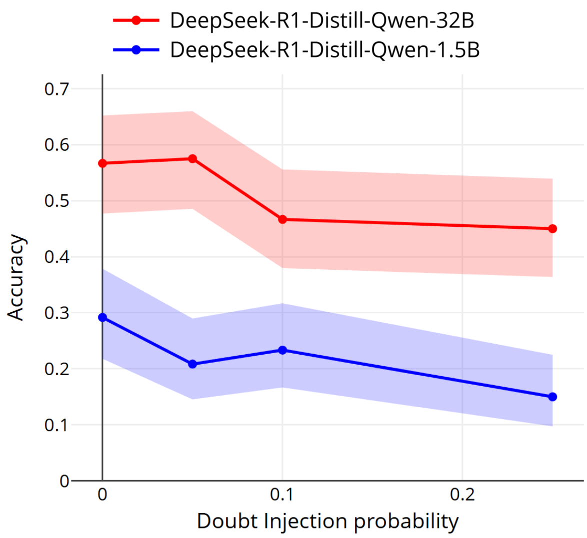

Figure 13: Accuracy of doubt injection string "But" on the AIME 2024 benchmark at temperature 0.6. 4 runs were made.

4 Conclusions

4.1 Future Work

References

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James Glass, and Pengcheng He. 2023. Dola: Decoding by contrasting layers improves factuality in large language models. *arXiv preprint arXiv:2309.03883*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2022. Contrastive decoding: Open-ended text generation as optimization. *arXiv preprint arXiv:2210.15097*.

Alisa Liu and Jiacheng Liu. 2023. The memotrap dataset. https://github.com/inverse-scaling/prize/blob/main/data-release/README.md. GitHub repository.

MAA. 2024. American invitational mathematics examination - aime. https://maa.org/student-programs/amc/. American Invitational Mathematics Examination - AIME 2024.

Philip and Hemang. 2024. Simplebench: The text benchmark in which unspecialized human performance exceeds that of current frontier models. https://simple-bench.com/. Retrieved from https://simple-bench.com/.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Weijia Shi, Xiaochuang Han, Mike Lewis, Yulia Tsvetkov, Luke Zettlemoyer, and Wen-tau Yih. 2024. Trusting your evidence: Hallucinate less with context-aware decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 783–791.

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*.

Han Wang, Archiki Prasad, Elias Stengel-Eskin, and Mohit Bansal. 2024. Adacad: Adaptively decoding to balance conflicts between contextual and parametric knowledge. *arXiv preprint arXiv:2409.07394*.

Xiaowei Yuan, Zhao Yang, Yequan Wang, Shengping Liu, Jun Zhao, and Kang Liu. 2024. Discerning and resolving knowledge conflicts through adaptive decoding with contextual information-entropy constraint. *arXiv preprint arXiv:2402.11893*.

A Prompt Templates

A.1 CAD Prompt Templates

Table 5: Prompt templates of the original CAD paper (Shi et al., 2024). "{context}" and "{question}" denote the where specific context $\mathbf{c}$ and prompt $\mathbf{x}$ (as in section 2.2.1) are inserted.

Prompt for $\mathrm{P}(y_t \mid \mathbf{c}, \mathbf{x}, \mathbf{y}_{<t})$**:**
{context} {question}
Prompt for $\mathrm{P}(y_t \mid \mathbf{x}, \mathbf{y}_{<t})$**:**
{question}

Table 6: Prompt templates of the COIECD paper (Yuan et al., 2024). "{context}" and "{question}" denote the where specific context $\mathbf{c}$ and prompt $\mathbf{x}$ (as in section 2.2.1) are inserted.

Prompt for $\mathrm{P}(y_t \mid \mathbf{c}, \mathbf{x}, \mathbf{y}_{<t})$**:**
Given the following information: {context} Answer the following question based on the given information with one or few words:{question} Answer:
Prompt for $\mathrm{P}(y_t \mid \mathbf{x}, \mathbf{y}_{<t})$**:**
Answer the following question based on your internal knowledge with one or few words: {question} Answer:

Table 7: Prompt templates of the AdaCAD paper (Wang et al., 2024). "{context}" and "{question}" denote the where specific context $\mathbf{c}$ and prompt $\mathbf{x}$ (as in section 2.2.1) are inserted.

Prompt for $\mathrm{P}(y_t \mid \mathbf{c}, \mathbf{x}, \mathbf{y}_{<t})$**:**
{context} Using only the references listed above, answer the following question: Question: {question}. Answer:
Prompt for $\mathrm{P}(y_t \mid \mathbf{x}, \mathbf{y}_{<t})$**:**
Answer the following question: Question: {question}. Answer:

Table 8: CAD and Additive CAD Prompt template used in this paper. "{context}" and "{question}" denote the where specific context $\mathbf{c}$ and prompt $\mathbf{x}$ (as in section 2.2.1) are inserted.

Prompt for $\mathrm{P}(y_t \mid \mathbf{c}, \mathbf{x}, \mathbf{y}_{<t})$**:**
Instruction: read the given information and answer the corresponding question. {context} Q: {question} A:
Prompt for $\mathrm{P}(y_t \mid \mathbf{x}, \mathbf{y}_{<t})$**:**
Instruction: answer the corresponding question. Q: {question} A:

A.2 Doubt Injection Prompts

Table 9: Full input prompt for the altered river crossing problem.

Full input prompt:
A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer. The reasoning process and answer are enclosed within <think> </think> and <answer> </answer> tags, respectively, i.e., <think> reasoning process here </think> <answer> answer here </answer>. User: Solve the following puzzle: A man and a goat are on the left side of a river. There is a wolf and a cabbage on the right side of the river. The man has a boat. The boat can carry only the farmer and a single item. How can the farmer get the goat to the right side of the river? Assistant:

Table 10: Prompt template for the SimpleBench test. "{question}" denotes the location of where the specific SimpleBench question is inserted.

Prompt template:
A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer. The reasoning process and answer are enclosed within <think> </think> and <answer> </answer> tags, respectively, i.e., <think> reasoning process here </think> <answer> answer X here </answer> where X is one of the letters A, B, C, D, E, or F. Do not include additional formatting in the final answer. User: {question} Assistant:

Table 11: Prompt template for the AIME 2024 benchmark test. Note that the begin-of-sentence, user and assistant tokens provided here aren't the exact unicode representations of those tokens. "{question}" denotes the location of where the specific AIME 2024 question is inserted.

Prompt template:
< | begin_of_sentence | >< | User | >You will be given a problem. Please reason step by step, and put your final answer within \boxed{}: {question}< | Assistant | ><think>