The Effect of Sampling Temperature on Problem Solving in Large Language Models

Matthew Renze [1] **Erhan Guven** [1]

Abstract

In this research study, we empirically investigate the effect of sampling temperature on the performance of Large Language Models (LLMs) on various problem-solving tasks. We created a multiple-choice question-and-answer (MCQA) exam by randomly sampling problems from standard LLM benchmarks. Then, we used four popular LLMs with five prompt-engineering techniques to solve the MCQA problems while increasing the sampling temperature from 0.0 to 1.0. Despite anecdotal reports to the contrary, our empirical results indicate that changes in temperature in the range 0.0 to 1.0 do not have a statistically significant impact on LLM performance for problem-solving tasks. In addition, these results appear to hold regardless of the LLM, the prompt-engineering technique, or the problem domain. All code, data, and supplemental materials are available on GitHub at: `https://github.com/matthewrenze/jhu-llm-temperature`.

1. Introduction

1.1. Background

In recent years, Large Language Models (LLMs) have revolutionized the field of artificial intelligence. Open-source LLMs and pay-per-use APIs have allowed engineers to incorporate LLMs in their AI systems. However, prompt engineering and hyperparameter tuning are required to work effectively with LLMs.

Prompt-engineering techniques help LLMs solve complex problems, avoid hallucinations, and provide more accurate responses. For example, we can use techniques like chain-of-thought, tree-of-thought, self-criticism, and self-consistency to improve LLM performance (Mialon et al., 2023; White et al., 2023).

[1]Johns Hopkins University, Baltimore, Maryland, USA.

In addition, several inference hyperparameters can be adjusted to change the LLM's output at runtime. For example, hyperparameters like sampling temperature, top-k sampling, repetition penalty, and maximum token length all affect the LLMs output and performance (OpenAI, 2023a; Touvron et al., 2023; Wang et al., 2023a).

Despite significant interest in LLMs and progress in LLM best practices, many open questions remain about optimal prompt-engineering techniques and inference hyperparameters for LLMs. To complicate matters, various local optima may exist for each problem type and problem domain (Wang et al., 2023a).

The prompt-engineering community has an abundance of opinions and anecdotal evidence regarding optimal prompt-engineering techniques and inference hyperparameter settings. However, we currently lack systematic studies and empirical evidence to support many of these claims.

As a result, this paper aims to address the open question of the optimal LLM sampling temperature for problem-solving tasks. In addition, we aim to provide a systematic study with empirical results to add to the growing body of knowledge used to create LLM and prompt-engineering best practices.

1.2. Sampling Temperature

Sampling temperature is a hyperparameter of an LLM used in a temperature-based sampling process. It controls the randomness of the model's output at inference time (Ackley et al., 1985; Hinton et al., 2015; Wang et al., 2020; 2023a).

At each step of an LLM's decoding process, the LLM uses the previous tokens to choose the next output token. The final layer of the LLM uses a softmax function to convert raw scores (logits) into probabilities.

In greedy sampling, the model will always choose the most likely next token. However, for probabilistic sampling, the next token is selected from a probability distribution.

Temperature sampling is a modification to the softmax function, which adjusts the resulting probability mass functions. In this modified softmax function, v_k is the k-th vocabulary token, l_k is the token's logit, and τ is a constant temperature. See equation 1.

$$\Pr(v_k) = \frac{e^{l_k/\tau}}{\sum_i e^{l_i/\tau}} \quad (1)$$

A lower temperature makes the output of the LLM more deterministic, thus favoring the most likely predictions. This conservativeness is captured by the model's tendency to produce more repetitive, focused, and less diverse output based on the patterns most commonly seen in the training data (Hinton et al., 2015; Wang et al., 2020; 2023a).

A higher temperature makes the output more random, thus favoring more "creative" predictions. This creativity is captured by the model's willingness to explore more unconventional and less likely outputs. Higher temperatures can lead to novel text, diverse ideas, and creative solutions to problems (Hinton et al., 2015; Wang et al., 2020; 2023a).

In the context of problem-solving, temperature can be seen as a trade-off between exploring and exploiting possible solutions within the solution space. Lower temperatures tend to exploit more probable solutions; higher temperatures explore the solution space more broadly.

1.3. Choosing a Sampling Temperature

Within the prompt-engineering community, there are a variety of opinions and best practices regarding the ideal sampling temperature for various problem-solving tasks (Microsoft; Shieh).

Low sampling temperatures are recommended for tasks requiring precision and factual accuracy, such as technical writing, code generation, or question-answering (Xu et al., 2022; Zhu et al., 2023). However, higher temperatures are recommended for tasks requiring creativity, such as writing poetry, creating stories, or brainstorming.

Higher temperatures also increase the probability of model hallucination. Hallucination is a phenomenon where an LLM produces statistically probable responses that are factually incorrect or nonsensical. As a result, optimal temperature selection is also a balance between creativity and hallucination (Lee, 2023).

Practical guidelines for choosing a sampling temperature for a specific task or problem domain are often vague or anecdotal. Prompt engineering guides often provide hypothetical examples of optimal sample temperatures for various tasks. However, they rarely cite any sources or provide empirical evidence[1].

As a result, the current state of choosing the optimal sample temperature for specific problems is largely based on guesswork, gut instinct, non-systematic experimentation, and iterative refinement[2].

1.4. Hypothesis

Given the lack of strong evidence for best practices regarding optimal sampling temperature, our research study empirically investigated the effect of LLM sampling temperature on problem-solving tasks. To answer our primary research question, we formulated the following hypotheses:

Null Hypothesis (H_0): Increasing or decreasing sampling temperature in the range 0.0 to 1.0 has no effect on the problem-solving performance of the LLM.

Alternative Hypothesis (H_1): Increasing or decreasing sampling temperature in the range 0.0 to 1.0 improves the problem-solving performance of the LLM agent.

To test these hypotheses, we measured the LLM's accuracy in correctly answering 1,000 multiple-choice questions across ten problem domains, ten times each. We then conducted a hypothesis test using a significance level of 0.05.

2. Methods

2.1. Data

The test dataset used in this research study consists of a series of Multiple-Choice Question-and-Answer (MCQA) exams derived from commonly used LLM performance benchmarks.

First, we reviewed the prior literature to identify benchmarks frequently used to evaluate LLMs. We limited our candidate benchmarks to those containing MCQA problems so that we could use correct-answer accuracy as our primary performance metric.

Next, we selected a set of problems that covered a range of problem domains (e.g., math, science, law, etc.) In addition, we selected problem sets that spanned a range of difficulty levels (e.g., secondary school, university, etc.) These problem sets can be seen in Table 1.

Then, we converted the benchmark problems from their original data format into a standardized data structure using the JSON Lines (JSON-L) format (Ward). Our standardized set of exams allowed us to use the exams interchangeably without modifying the code in the test harness. See Appendix B - Figure 8 for a sample of an MCQA problem.

Finally, we created two MCQA exams of different sizes.

[1] A few empirical studies exist that indicate sampling temperature does have an effect on LLM performance on some types of problem-solving tasks (e.g., code generation, engineering exams, etc.) (Xu et al., 2022; Pursnani et al., 2023; Zhu et al., 2023)

[2] Even the GPT-4 Technical Report explains that the authors used their "best-guess" when choosing sampling temperatures while evaluating GPT-4 on various benchmarks. See Appendix A in the GPT-4 Technical Report (OpenAI, 2023c).

Table 1. Source of problem sets used to create the multi-domain MCQA exam.

Problem Set	Benchmark	Domain	Questions	Source
ARC Challenge Test	ARC	Science	1,173	(Clark et al., 2018)
AQUA-RAT	AGI Eval	Math	254	(Zhong et al., 2023)
Hellaswag Val	Hellaswag	Common Sense Reasoning	10,042	(Zellers et al., 2019)
LogiQA (English)	AGI Eval	Logic	651	(Zhong et al., 2023; Liu et al., 2020)
LSAT-AR	AGI Eval	Law (Analytic Reasoning)	230	(Zhong et al., 2023; Wang et al., 2021)
LSAT-LR	AGI Eval	Law (Logical Reasoning)	510	(Zhong et al., 2023; Wang et al., 2021)
LSAT-RC	AGI Eval	Law (Reading Comprehension)	260	(Zhong et al., 2023; Wang et al., 2021)
MedMCQA Valid	MedMCQA	Medicine	6,150	(Pal et al., 2022)
SAT-English	AGI Eval	English	206	(Zhong et al., 2023)
SAT-Math	AGI Eval	Math	220	(Zhong et al., 2023)

we created a large exam with 1,000 questions by randomly sampling 100 problems from each of the ten problem sets. This 1,000-question (large) exam was used with GPT-3.5 to perform our hypothesis test and a detailed analysis of temperature across problem domains and prompt types.

However, we also created a smaller exam of 100 questions by randomly sampling ten questions from each of the ten domain-specific problem sets. This 100-question (small) exam was used for our high-level analysis of sampling temperature across all four models[3].

2.2. Models

The LLM models used in this research project consist of four commonly used foundational LLMs. To complement our analysis, we also performed experiments using five prompts created with commonly used prompt-engineering techniques.

First, we reviewed the prior literature to identify candidate LLMs commonly used for problem-solving tasks. We limited our candidate models to those that allowed the model's sampling temperature to be specified via their API. These LLMs included GPT-3.5, GPT-4, Llama 2 7B, and Llama 2 70B (OpenAI, 2022; 2023b;c; Touvron et al., 2023).

Next, we reviewed the existing literature for commonly used prompt-engineering techniques. We limited our candidate prompts to those that could be performed in a single request-and-response cycle with one-shot in-context learning. We excluded multi-step agents, few-shot learning, and model fine-tuning.

As a result, we selected the following five prompt-engineering techniques to construct our system prompts:

- **Baseline** - no prompt engineering; the LLM is instructed to return only a single multiple-choice answer as its output (e.g., 'Answer("C")').
- **Domain Expertise** – the system prompt specifies that the LLM is an expert in the problem domain of the exam (e.g., "medicine") or the topic of the problem (e.g., "anatomy") (White et al., 2023).
- **Self-recitation** – the system prompt instructs the LLM to recite its own internal knowledge about the problem before answering the question (Sun et al., 2023; White et al., 2023).
- **Chain-of-Thought (CoT)** – the system prompt instructs the LLM to "think step-by-step" to encourage it to reason through the problem procedurally (Kojima et al., 2022; Wei et al., 2022).
- **Self-criticism** – the system prompt instructs the LLM to critically evaluate its previous thoughts to identify any errors in facts, logic or reasoning (Huo et al., 2023; Wang et al., 2023b).
- **Composite** – the system prompt combines domain expertise, self-recitation, chain-of-thought, and self-criticism.

Finally, we provided the LLM with a single example problem-and-solution pair for one-shot in-context learning. The example solution was adapted for each prompt based on the prompt-engineering technique used. For example, the chain-of-thought prompt included a chain-of-thought in its solution. See Appendix A - Figure 7 for a sample prompt.

2.3. Metrics

To test our hypothesis, we measured the LLM's correct-answer accuracy as our primary performance metric. To further support our findings, we also measured the similarity of the LLM's responses using a series of text-similarity metrics.

These metrics are defined as follows:

- **Accuracy** – the ratio of correctly answered MCQA problems to the total number of problems attempted

[3]We used the smaller 100-question exam for GPT-4, Llama 7B and Llama 70B due to cost considerations.

- **Jaccard similarity** – the ratio of the intersection to the union of word sets in the output text (Jaccard, 1912)
- **Bag-of-Words (BoW) similarity** – comparison of the frequency of each word, ignoring order (Harris, 1954)
- **TF-IDF similarity** – comparison of word frequency weighted by inverse document frequency (Jones, 1972)
- **Levenshtein similarity** – the number of edits needed to change one text into the other (Levenshtein, 1966)
- **BLEU score** – comparison of similarity based on n-gram overlap (Papineni et al., 2001)
- **SBERT similarity** – semantic similarity computed using Sentence-BERT embeddings (Reimers & Gurevych, 2019)

2.4. Analysis

We performed an analysis of correct-answer accuracy across temperatures ranging from 0.0 to 1.0. in increments of 0.1 using GPT-3.5 with the CoT prompt and the 1,000-question exam. We then performed a Kruskal-Wallis test to evaluate the statistical significance of any changes in accuracy as a function of temperature[4]. We evaluated our hypotheses at a significance level $\alpha = 0.05$.

3. Results

3.1. Accuracy vs. Temperature

Our analysis revealed that the mean accuracy remained relatively stable across all sampling temperatures.

The Kruskal-Wallis test for GPT-3.5 on the 1,000-question exam yielded the following results:

- **Baseline:** $H(10) = 0.800, p = 1.000$
- **Domain Expertise:** $H(10) = 2.508, p = 0.991$
- **Self-Recitation:** $H(10) = 4.592, p = 0.917$
- **Chain-of-Thought:** $H(10) = 10.407, p = 0.406$
- **Composite:** $H(10) = 9.454, p = 0.490$

Given a significance level of 0.05, the higher p-values indicate that the differences in accuracy across temperatures were not statistically significant. Thus, we did not find sufficient evidence to reject the null hypothesis.

In addition, we saw similar results for GPT-4 using the smaller 100-question exam. The two Llama models did not perform better than statistically random chance on the 100-question exam, so their results had to be excluded from our analysis[5]. See Figure 2.

These results suggest that changes to sampling temperature in the range of 0.0 to 1.0 do not have a statistically significant effect on LLM performance in problem-solving. However, visual inspection of GPT-3.5 accuracy with 1,000 questions indicates a non-significant decrease in performance as a

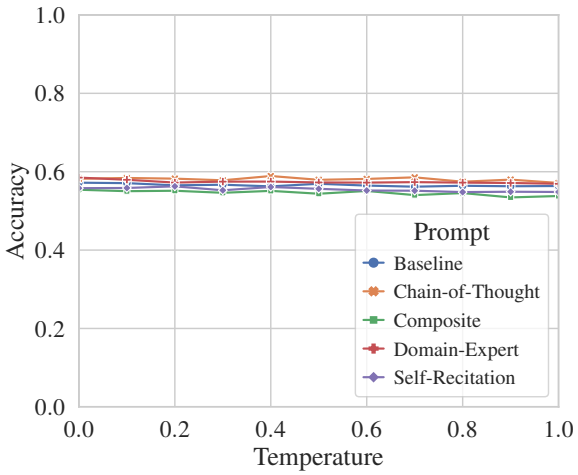

Figure 1. Accuracy by temperature and prompt for GPT-3.5 with 1,000 questions. Performance remains relatively stable across all temperatures and prompts. However, there is a non-significant decrease in performance as a function of temperature.

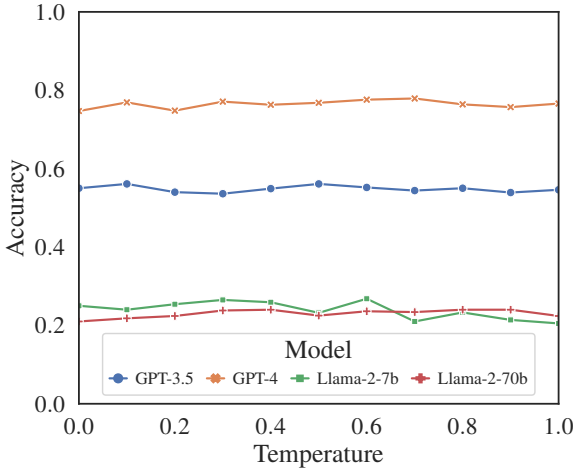

Figure 2. Accuracy by temperature and model. Performance remains stable across sampling temperatures for all four LLMs on the 100-question MCQA exam. However, both Llama 2 models performed no better than statistically random guesses.

[4]We used Kruskal-Wallis because the data (i.e., correct-answer accuracy by question) were not normally distributed. Rather, they were bimodally distrbuted with centers at 0.0 and 1.0.

[5]The low accuracy of the Llama models was the result of it producing incorrectly formatted answers (7B: 38%; 70B: 57%) and correctly formatted but incorrect answers (7B: 57%; 70B: 19%).

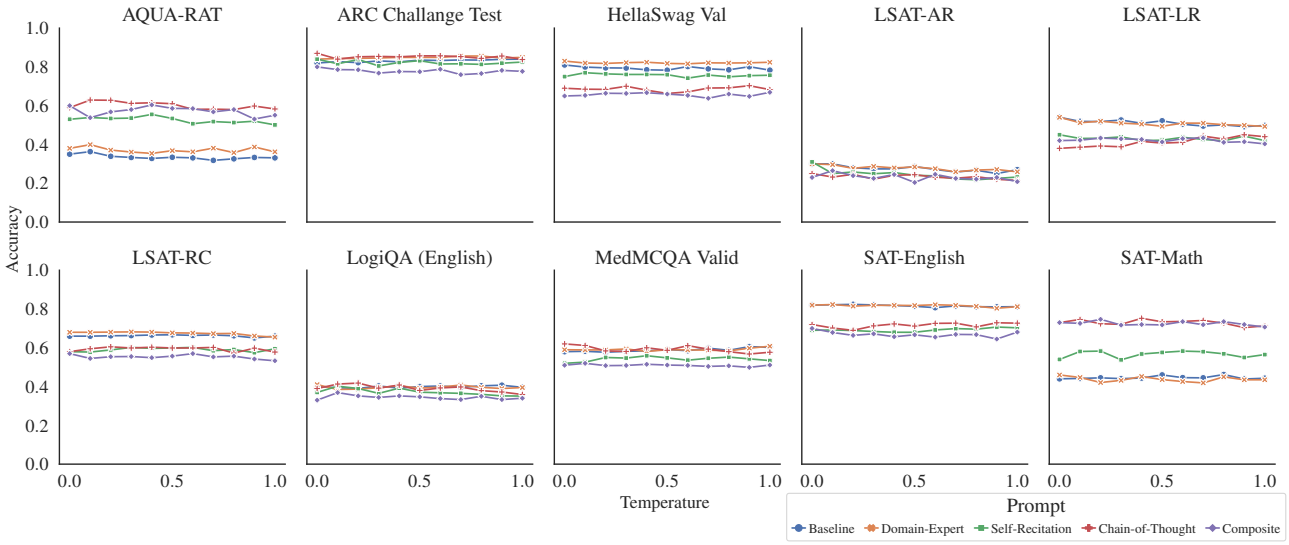

Figure 3. Accuracy of GPT-3.5 by temperature, prompt, and exam. Performance remains relatively stable across all temperatures regardless of the system prompt or the problem domain for GPT-3.5 on all 1,000 multiple-choice questions.

function of temperature – which matches our intuition about randomness in sampling temperature. See Figure 1.

Visual examination of the performance of GPT-3.5 across prompts and problem domains also supported these findings. We cannot identify any discernible trends in correct-answer accuracy for GPT-3.5 as a function of temperature when segmented by prompt type or exam. See Figure 3.

However, beyond a temperature of 1.0, problem-solving performance begins to decrease significantly starting at a temperature of 1.2 until the model produces statistically random guesses at a temperature of 1.4. These results also match our intuition of randomness in sampling temperature. See Figure 4.

3.2. Text Variability vs. Temperature

To further support our results, we analyzed text variability as a function of temperature. Our results show a clear trend of decreasing text similarity (thus increasing text variability) as temperature increases. This confirms that the sampling-temperature hyperparameter was behaving as expected.

These results are consistent with our understanding of sampling temperature, indicating that higher temperatures produce more widely varied outputs. In addition, these results hold regardless of the LLM model, prompt-engineering technique, or problem domain. See Figures 5 and 6.

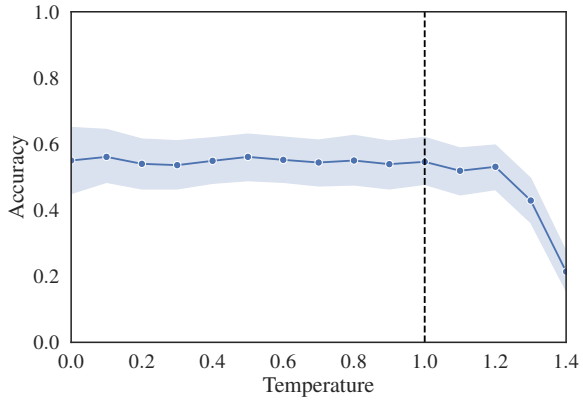

Figure 4. Accuracy by temperature for GPT-3.5 with 100 questions beyond a temperature of 1.0. Performance begins to drop significantly at a temperature of 1.2 until performance becomes no better than statistically random guesses at 1.4.

4. Discussion

4.1. Limitations

There were several limitations to our research study:

First, our study was limited to a small subset of problems, problem domains, and problem-solving tasks. As a result, our findings may not hold for larger data sets, different problem domains, or other types of problem-solving tasks.

Second, we only explored a small subset of prompt-

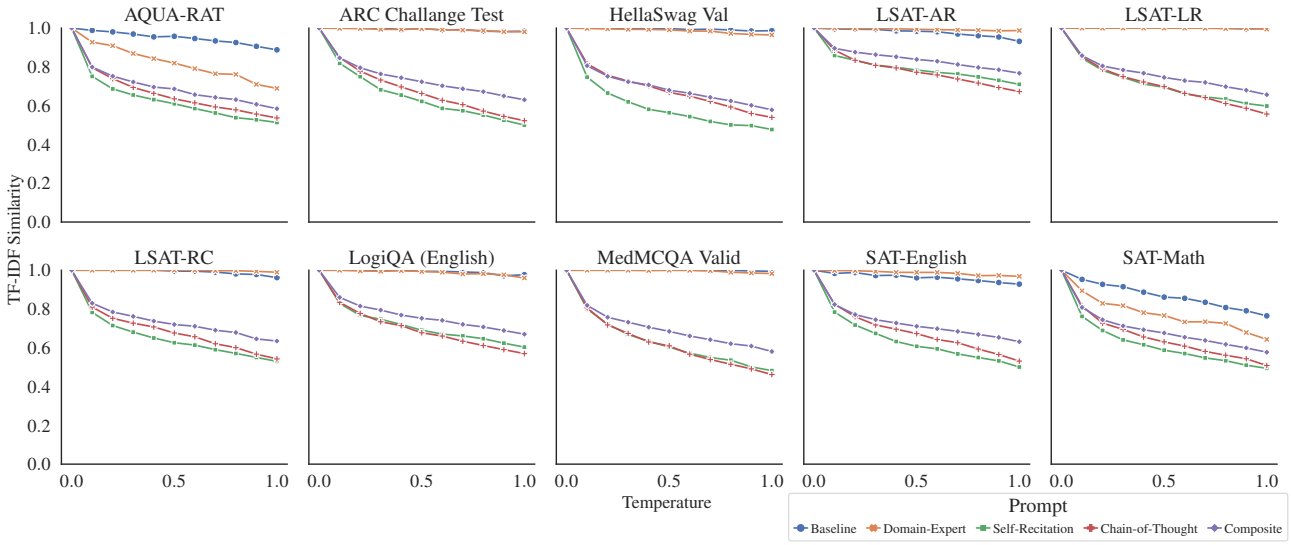

Figure 5. Text similarity by temperature, prompt, and exam - TF-IDF text similarity decreases as temperature increases regardless of the prompt or problem domain. Note that the baseline and domain-expert prompts produce short LLM responses, so their decrease in text similarity across temperatures is minimal. However, math problems frequently cause the LLM to automatically produce a CoT.

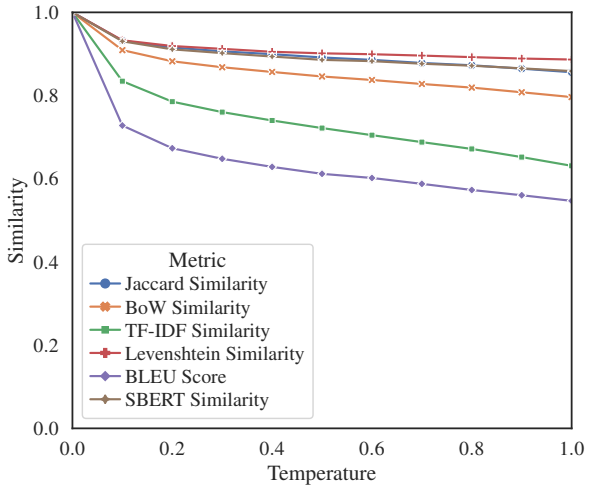

Figure 6. Text similarity by temperature and metric. Text similarity decreases as temperature increases for GPT-3.5 on all 1,000 questions using all six text-similarity metrics.

engineering techniques using a single prompt-and-response cycle with one-shot in-context learning. As a result, there may be more complex prompts or agent architectures that better leverage sampling temperature in their problem-solving capabilities.

Third, due to cost considerations, we could only fully explore the landscape of system prompts and problem domains with GPT-3.5. As a result, there may exist other combinations of system prompts and problem domains where other LLMs may better utilize sampling temperature for problem-solving.

4.2. Implications

This research study provides empirical evidence that changes to sampling temperature in the range of 0.0 to 1.0 do not affect the problem-solving capabilities of LLMs on MCQA problems.

Answering this question could save AI engineers significant time and resources evaluating LLM solutions with various sampling temperatures. In addition, it may reduce unproductive debates in the prompt-engineering community about optimal sampling temperatures for problem-solving tasks.

This research also provides more general insight for AI researchers studying model hallucination and state-space search with LLMs. Our results show that increasing LLM temperature up to 1.0 does not create hallucinations that lead to incorrect MCQA solutions. In addition, higher temperatures do not appear to improve MCQA solution-space search in ways that lead to correct solutions more often.

4.3. Future Research

To improve upon this research, we propose the following follow-up experiments:

First, we should perform additional experiments with more

MCQA problems and problem domains. We recommend specifically targeting tasks and problem domains that require more creative solutions or lateral "out-of-the-box" thinking.

Second, we recommend expanding beyond MCQA problems to other types of problem-solving tasks whose correct answers are more open-ended. The limited effects of sampling temperature in our experiments may have simply resulted from the constraints imposed by the structure of MCQA problems.

Third, we recommend performing these experiments with additional LLMs. Other proprietary and open-source LLMs may utilize temperature in ways that benefit their specific models but did not benefit GPT-3.5 and GPT-4.

Finally, we recommend a more in-depth error analysis to determine if any sub-types of problems within these problem domains benefit from changes to sampling temperature. It is possible that statistical noise or averaging may have hidden individual problems that were sensitive to changes in sampling temperature.

5. Conclusion

This research study provided an empirical investigation of the effect of sampling temperature on the problem-solving performance of LLMs across multiple problem domains.

We demonstrated that changes in sampling temperature within the range of 0.0 to 1.0 do not produce statistically significant differences in problem-solving performance on MCQA problems across multiple LLMs, prompt-engineering techniques, and problem domains.

These results have practical implications for AI engineers using LLMs to create new AI systems. In addition, they have theoretical implications for AI researchers studying model hallucination and solution-space search with LLMs.

Impact Statement

This paper presents work whose goal is to advance the field of Artificial Intelligence and Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically addressed here.

References

Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9: 147–169, 1985. ISSN 0364-0213. doi: https://doi.org/10.1016/S0364-0213(85)80012-4.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *ArXiv*, 3 2018.

Harris, Z. S. Distributional structure. *WORD*, 10:146–162, 8 1954. ISSN 0043-7956. doi: 10.1080/00437956.1954.11659520.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv*, 3 2015.

Huo, S., Arabzadeh, N., and Clarke, C. L. A. Retrieving supporting evidence for generative question answering. *arXiv*, 9 2023. doi: 10.1145/3624918.3625336.

Jaccard, P. The distribution of flora in the alpine zone. *New Phytologist*, 11:37–50, 2 1912. ISSN 0028-646X. doi: 10.1111/j.1469-8137.1912.tb05611.x.

Jones, K. S. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21, 1 1972. ISSN 0022-0418. doi: 10.1108/eb026526.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. Large language models are zero-shot reasoners. volume 35, pp. 22199–22213, 5 2022.

Lee, M. A mathematical investigation of hallucination and creativity in GPT models. *Mathematics*, 11:2320, 5 2023. ISSN 2227-7390. doi: 10.3390/math11102320.

Levenshtein, V. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10:707–710, 1966.

Liu, J., Cui, L., Liu, H., Huang, D., Wang, Y., and Zhang, Y. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. 2020.

Mialon, G., Dessì, R., Lomeli, M., Nalmpantis, C., Pasunuru, R., Raileanu, R., Rozière, B., Schick, T., Dwivedi-Yu, J., Celikyilmaz, A., Grave, E., LeCun, Y., and Scialom, T. Augmented language models: a survey. *arXiv*, 2 2023.

Microsoft. Completions - learn how to generate or manipulate text. URL https://learn.microsoft.com/en-us/azure/ai-services/openai/how-to/completions.

OpenAI. Introducing ChatGPT, 11 2022. URL https://openai.com/blog/chatgpt.

OpenAI. OpenAI - API reference, 2023a. URL https://platform.openai.com/docs/api-reference/chat/create.

OpenAI. GPT-4, 3 2023b. URL https://openai.com/research/gpt-4.

OpenAI. GPT-4 technical report. *arXiv*, 3 2023c. URL https://arxiv.org/abs/2303.08774.

Pal, A., Umapathi, L. K., and Sankarasubbu, M. MedMCQA: A large-scale multi-subject multi-choice dataset for medical domain question answering. pp. 248–260. PMLR, 2022.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. BLEU. pp. 311. Association for Computational Linguistics, 2001. doi: 10.3115/1073083.1073135.

Pursnani, V., Sermet, Y., and Demir, I. Performance of ChatGPT on the US fundamentals of engineering exam: Comprehensive assessment of proficiency and potential implications for professional environmental engineering practice. *arXiv*, 4 2023.

Reimers, N. and Gurevych, I. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. Association for Computational Linguistics, 8 2019.

Shieh, J. Best practices for prompt engineering with OpenAI API. URL https://help.openai.com/en/articles/6654000-best-practices-for-prompt-engineering-with-openai-api.

Sun, Z., Wang, X., Tay, Y., Yang, Y., and Zhou, D. Recitation-augmented language models. 10 2023.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models. *arXiv*, 7 2023.

Wang, C., Liu, S. X., and Awadallah, A. H. Cost-effective hyperparameter optimization for large language model generation inference, 2023a.

Wang, P.-H., Hsieh, S.-I., Chang, S.-C., Chen, Y.-T., Pan, J.-Y., Wei, W., and Juan, D.-C. Contextual temperature for language modeling. *arXiv*, 12 2020.

Wang, R., Wang, H., Mi, F., Chen, Y., Xu, R., and Wong, K.-F. Self-critique prompting with large language models for inductive instructions. *arXiv*, 5 2023b.

Wang, S., Liu, Z., Zhong, W., Zhou, M., Wei, Z., Chen, Z., and Duan, N. From lsat: The progress and challenges of complex reasoning. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 30:2201–2216, 8 2021.

Ward, I. JSON lines. URL https://jsonlines.org/.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. *arXiv*, 1 2022.

White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., and Schmidt, D. C. A prompt pattern catalog to enhance prompt engineering with ChatGPT. *arXiv*, 2 2023.

Xu, F. F., Alon, U., Neubig, G., and Hellendoorn, V. J. A systematic evaluation of large language models of code. pp. 1–10. Association for Computing Machinery, 2022. ISBN 9781450392730. doi: 10.1145/3520312.3534862.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. HellaSwag: Can a machine really finish your sentence? 2019.

Zhong, W., Cui, R., Guo, Y., Liang, Y., Lu, S., Wang, Y., Saied, A., Chen, W., and Duan, N. AGIEval: A human-centric benchmark for evaluating foundation models. *ArXiv*, 4 2023.

Zhu, Y., Li, J., Li, G., Zhao, Y., Li, J., Jin, Z., and Mei, H. Improving code generation by dynamic temperature sampling. *arXiv*, 9 2023.

A. Prompts

```
[System Prompt]
You are an expert in {{expertise}}.
Your task is to answer the following multiple-choice questions.
First, you should recite all of the relevant knowledge you have about the question and each option.
Next, you should think step-by-step through the problem to ensure you have the correct answer.
Then, you should critically evaluate your thoughts to identify any flaws in your facts, logic, and reasoning.
Finally, you MUST answer the question using the following format 'Action: Answer("[choice]")'
The parameter [choice] is the letter or number of the answer you want to select (e.g. "A", "B", "C", or "D")
For example, 'Answer("C")' will select choice "C" as the best answer.
The answer MUST ALWAYS be one of the available choices; it CANNOT be "None of the Above".
If you think the answer is "none of the above", then you MUST select the most likely answer.

[Example Problem]
Question: What is the capital of the state where Johns Hopkins University is located?
Choices:
  A: Baltimore
  B: Annapolis
  C: Des Moines
  D: Las Vegas

[Example Solution]
Knowledge:
  Johns Hopkins University is located in Baltimore, Maryland.
  A: Baltimore is a city located in the state of Maryland, but it is not the capital of Maryland.
  B: Annapolis is a the capital of the State of Maryland.
  C: Des Moines is a city located in the State of Iowa, but it is not the capital of Iowa.
  D: Las Vegas is located in the State of Nevada, but it is not the capital of Nevada.
Thought:
  Johns Hopkins University is located in Baltimore.
  Baltimore is a city located in the state of Maryland.
  The capital of Maryland is Baltimore.
  Therefore, the capital of the state where Johns Hopkins University is located is Baltimore.
  The answer is A: Baltimore.
Criticism:
  You are correct that Johns Hopkins is located in the State of Baltimore.
  However, the capital of Maryland is Annapolis, not Baltimore.
  So, the correct answer is actually B: Annapolis.
Action: Answer("B")
```

Figure 7. Sample of the composite system prompt with a one-shot example (i.e., problem-and-solution pair).

B. Data

```
{
  "source": "arc/arc-challenge-test",
  "source_id": 1,
  "topic": "Science",
  "context": "",
  "question": "An astronomer observes that a planet rotates faster
              after a meteorite impact. Which is the most likely effect
              of this increase in rotation?",
  "choices": {
    "A": "Planetary density will decrease.",
    "B": "Planetary years will become longer.",
    "C": "Planetary days will become shorter.",
    "D": "Planetary gravity will become stronger." },
  "answer": "C",
  "solution":""
}
```

Figure 8. Sample of an MCQA problem in JSON-L format – with whitespace added for readability.