RESEARCH ARTICLE

Benchmark-based evaluation and optimization of protein Large Language Models: Test bank construction and parameter tuning analysis

Wenjun Yang[1, 2], Beow Keat Yap[3], Xinsheng Liu[2], Theam Foo Ng[4], Yoke Seng Wong[1], Bingkun Ren[2], Shir Li Wang[1, 5, 6, *]

[1]Faculty of Computing and Meta-Technology, University Pendidikan Sultan Idris, Tanjong Malim, Perak, Malaysia. [2]School of Computer and Information Engineering, Heihe University, Heihe, Heilongjiang, China. [3]School of Pharmaceutical Sciences, [4]Centre for Global Sustainability Studies, University Sains Malaysia, Minden, Penang, Malaysia. [5]Data Intelligence and Knowledge Management (DILIGENT), [6]Robotics and Cyber-Physical System (RACS), University Pendidikan Sultan Idris, Tanjong Malim, Perak, Malaysia.

Received: November 26, 2025; **accepted:** March 26, 2026.

With the rapid development of natural language processing technology, Large Language Models have shown great application potential in biomedical fields, particularly in protein-related research. However, the scientific community currently lacks standardized and diverse protein-specific test sets for model evaluation, and systematic research on how parameter tuning affects the performance of Large Language Models in protein-related tasks remains insufficient. This research gap restricts the reliable application and performance improvement of such models in biomedical scenarios. This study constructed a standardized test question bank and explored the effects of key parameters on model performance, thereby providing a practical optimization framework for Large Language Models applied to protein-related tasks. The study constructed a high-quality and diverse protein-related test question bank, providing standardized testing data for objective model evaluation followed by extensive experiments to analyze the impacts of temperature, top_p, and presence penalty on four critical model performance metrics including precision, recall, cosine, and METEOR score under different parameter combination settings. The results showed that presence penalty exerted a significant regulatory effect on model performance with strictly positive presence penalty values that suppressed topic repetition led to volatile performance for specific temperature settings, while neutral or slightly negative presence penalty values yielded more consistent performance. Moderate temperature settings balanced output diversity and performance stability, whereas extreme temperature values resulted in suboptimal performance. Top_p values within the conventional range maintained stable model performance, but extreme values caused significant performance degradation, especially when paired with higher temperature settings. The optimal parameter configuration was identified as neutral or slightly negative presence penalty, temperature between 0.8 and 1.2, and top_p between 0.2 and 0.8, which effectively improved the model's precision, recall, semantic consistency, and text generation quality. The constructed protein-specific test question bank could provide standardized support for the evaluation of protein-focused Large Language Models, and the optimized parameter configuration could significantly enhance model performance in protein-related tasks. This research filled the existing gap in the construction of protein-specific test question banks and enriched systematic research on parameter tuning of Large Language Models in biomedical fields, providing a theoretical and practical basis for the further application and development of natural language processing technology in protein research and related biomedical fields. Protein language model benchmark and code can be accessed at https://github.com/w87027619332-cloud/newhub.git.

Keywords: protein large language model; test question bank; parameter tuning; benchmarking.

[*]**Corresponding author:** Shir Li Wang, Faculty of Computing and Meta-Technology, University Pendidikan Sultan Idris, Tanjong Malim, Perak 35900, Malaysia. Email: shirli_wang@meta.upsi.edu.my.

Introduction

In the field of protein research, leveraging Large Language Models to assist with protein structure prediction, functional analysis, and related knowledge mining holds broad promising applications [1]. In recent years, protein-based Large Language Models such as ESM-2 and ProtGPT2 have shown potential in sequence generation and structure prediction tasks, and databases such as UniProt and Protein Data Bank (PDB) have provided basic data support for model training. Meanwhile, parameter tuning strategies have been proven to regulate output diversity and accuracy in the field of natural language generation, but these results have not yet been systematically transferred to protein-related tasks, and related research is still in its early stages.

Current research on constructing protein-related test question repositories remains insufficient, primarily characterized by a lack of high-quality, diverse annotated datasets and a standardized, unified evaluation framework, which limits the systematic assessment and optimization of model performance and constrains the effectiveness of models in practical applications [2]. Meanwhile, to meet the diverse, coherent, and controllable content generation needs of different application scenarios, researchers have proposed various parameter adjustment strategies such as temperature [3], nucleus sampling (top_p) [4], and penalty mechanisms including presence_penalty and frequency_penalty [5]. Proper configuration of these parameters significantly impacts the diversity, consistency, and controllability of model outputs, yet there is a lack of systematic theoretical analysis and empirical research in this area. Lim *et al*. investigated the code generation parameters for ChatGPT 3.5 by focusing on the temperature setting to the default value of 1 to test the model's "typical behavior" and found that the model's outputs under this parameter were inherently nondeterministic [6]. However, because the top_p, frequency_penalty, and presence_penalty parameters were not studied, the impact of these three parameters on code repetition and diversity could not be evaluated, leading to a limitation in the scope of parameter exploration [7]. Wang *et al*. focused on the four categories of hyperparameters and found that both temperature (0 - 1) and top_p (0 - 1) controlled randomness and tended to conflict when tuned simultaneously. The frequency_penalty (default 0, range [-2, 2]) regulated word frequency, while the presence_penalty (default 0, range [-2, 2]) controlled the occurrence of specific content. Additionally, the complex interactions among hyperparameters made manual optimization difficult. The EcoOptiGen framework enabled joint tuning of these parameters and had been shown to be effective for ChatGPT [8]. Arora *et al*. targeted GPT-3.5 Turbo to investigate how four hyperparameter categories affected Python code generation. A total of 1,134 parameter combinations generated 14,742 code snippets that were subsequently classified and evaluated. The results indicated that temperature had the strongest effect with a strong negative correlation to correctness with optimal results occurred below 0.5. top_p also played a significant role with higher accuracy observed below 0.75. The frequency_penalty performed well within the range of -1 to 1.5, while the presence_penalty showed minimal influence on the output quality [9]. In addition, another study that also centered on GPT-3.5 Turbo confirmed that temperature and top_p had the most significant impact with a strong negative correlation to correctness. The frequency_penalty within the range of -1 to 1.5 helped reduce the production of inexecutable code, while the presence_penalty had minimal effect with slight improvements observed when above -1 [10]. In the context of power system simulation applications, Jia *et al*. specified the Large Language Model (LLM) parameter configuration as 0.1 for temperature to reduce randomness, 1.0 for top_p to utilize the full probability distribution, and 0 for both frequency_penalty and presence_penalty. Based on this setup, the multi-agent framework demonstrated excellent performance across 69

tasks, achieving high success rates, rapid execution, and low costs [11]. Shlomov *et al*. explored network intelligent agent and found that, when calling the GPT-4o application programming interface, both temperature and top_p were set to 1, while frequency_penalty and presence_penalty were set to 0. However, for the document object model grounding task, by using Llama-2-70b, the temperature was set to 0.1. The results showed that planning was the performance bottleneck, and performance improved after optimizing the ranking strategy [12].

This study systematically explored how to develop a benchmark testing system that was suitable for protein LLM and assessed the effects of different parameter settings on generation quality. The specific tasks of this study included constructing a high-quality protein-related test question bank through automated generation, quality control, and expert review, providing standardized evaluation data for models, analyzing the influence patterns of various parameters on model outputs to reveal their roles in content diversity and quality control, and conducting a series of experiments to evaluate the impact of parameter tuning on metrics including precision, recall, cosine, METEOR and propose optimized configuration schemes. The results of this research would provide a theoretical foundation and practical guidance for improving the performance of protein Large Language Models and promote their in-depth application in the life sciences.

Materials and methods

Protein data collection

The data of 300 proteins were collected from UniProt (https://www.uniprot.org/uniprotkb) [13], PDB (https://www.rcsb.org/) [14], and NCBI (https://static.pubmed.gov/portal/portal.fcgi/) [15], which included the information of entry, protein names, length, sequence, function, organism, subunit structure, tissue specificity, involvement in disease, post-translational modification, domain [CC], domain [FT], induction, DOI ID, PubMed ID, date of creation and formed the core material repository for subsequent question generation, ensuring that the questions focused on cutting-edge topics and key knowledge points in protein research, reflecting the latest research findings and scholarly developments in the field of protein science.

Depth and length compatibility

Question design should possess sufficient depth and complexity, avoiding overly simplistic content. The length of the answers should be controlled within a range of 150 - 200 words, allowing for a reasonable deviation of approximately ± 30 words. This length specification ensured that the answers could thoroughly explain the core points while maintaining a relatively uniform length standard, thereby preventing significant variations in answer length from affecting the precise assessment of response accuracy.

Formatting standardization and uniformity

Answers should be presented in coherent, complete paragraph form. The use of bullet points or list formats was strictly prohibited. This uniform formatting requirement helped eliminate potential disruptions caused by differences in answer presentation, making comparison and analysis between the model output and standard answers more direct and efficient. It also enhanced the scientific rigor and standardization of the evaluation process.

JavaScript Object Notation (JSON) format specification

Given that the experiment employed an automated answering process, the question format needed strictly adhere to JSON standards (https://datatracker.ietf.org/doc/html/rfc8259). During the formulation process, any format symbols that could potentially cause parsing errors such as "," should be strictly avoided. Strict adherence to JSON formatting standards was a crucial technical prerequisite to ensure the stable operation of the program and the accurate

processing of data.

Prompt clear goals

Based on the expected key points and academic focus of protein short-answer questions, the core objectives of the questions should be precisely defined to guide the generation of in-depth, knowledge-focused test content. The questions should closely revolve around advanced biological topics such as functional mechanisms of proteins, the intrinsic relationship between structure and function, post-translational modifications, signal transduction pathways, inter-molecular interactions, disease associations, or evolutionary significance. Ensuring that the prompt words effectively guided AI to focus on these key knowledge areas would generate questions and answers with higher academic value and depth.

Prompt formulation and optimization

Carefully craft the initial prompt to ensure it comprehensively and accurately conveyed the question requirements. Based on the acquired fundamental information about proteins, prompt words were designed to effectively guide AI to generate short-answer questions and standard answers centered on specific protein attributes such as subunit structural features, tissue-specific expression, disease association mechanisms, *etc*. During the formulation process, clear and concise language and precise instructions were focused on avoiding vague or ambiguous expressions.

Verification and iterative improvement of AI-generated results

The preliminary designed prompt words were input into the AI system and conducted an in-depth analysis and rigorous validation of its generated results. A comprehensive evaluation was performed across multiple dimensions including the accuracy, completeness, conciseness of the answers, and their alignment with the question objectives to determine whether they met the experimental requirements. Based on the evaluation outcomes, the prompt wording was selectively optimized and refined to improve performance.

Integrated application of data and prompt

A complete input information set was constructed by organically combining the collected and organized data of 300 proteins with the optimized prompt words. The response format requirements were clearly specified to ensure that the output was standardized and consistent.

The process of question generation and quality control

Kimi model was employed to combine pre-designed prompt words and protein data for question generation [16]. Based on deep learning and natural language processing techniques, the Kimi model possessed ultra-long context understanding and multimodal alignment abilities, enabling it to generate coherent and logically structured questions. During the question generation process, the quality of the output questions was closely monitored. If issues such as insufficient depth, formatting errors, or incomplete answers were observed, timely feedback was provided to the model and specified detailed modification requirements. After repeated verification and revision by biologists, the final generated questions would fully meet the established standards and possess a high level of academic rigor and testing value.

Standardized record and organization

A comprehensive review was conducted, and the questions generated by the Kimi model were filtered. The qualified questions were standardized according to a unified format, which ensured clear correspondence between questions and answers and facilitated easy reference.

Implementation of translation work

Kimi large model was used for question translation. Since the translation content was relatively concise and clear, there was no need to design specialized prompts. The standardized questions were simply input into the Kimi model to obtain the English translation results following

the previous standardization process to reorganize the translated questions, ensuring that the format and content remained consistent, standardized, and uniform.

JSON format conversion and integration

The translated English questions were transformed and organized according to the predetermined JSON format requirements. During the conversion process, Python scripts (https://www.python.org/) were utilized for automated batch processing to enhance efficiency and accuracy. Ultimately, a complete, standardized, and application-friendly collection of 300 questions along with their corresponding JSON files was created, which primarily focused on six major areas including protein structure-function mechanisms, metabolism and catalysis, signal transduction and regulation, immune and pathogen interactions, cellular life process regulation, and applications and drug development. These major areas covered a wide range of research frontiers from basic biology to medicine and biotechnology, providing a standardized, high-quality data resource to support subsequent question-answering evaluation, model performance testing, and related research work.

GPT-4o-mini and parameter introduction

GPT-4o-mini provided a rich set of tuning mechanisms with strong generalization ability, multi-task processing capacity, and flexible control mechanisms, allowing users to flexibly control output diversity and creativity to meet different scenario requirements.

(1) Top_p

Top_p controled the sampling process by selecting only the subset of vocabulary whose cumulative probability reached the threshold, thereby ensuring a certain level of diversity while avoiding the generation of unreasonable low-probability words. All candidate words were sorted in descending order by their probabilities p_w and denoted as w_1, w_2, ..., w_N with corresponding probabilities p_{w1}, p_{w2}, ..., p_{wN} and then the subset V_{topp} was chosen as follows.

$$\sum_{i=1}^{k} p_{w_i} \leq top\_p \quad (1)$$

$$V_{top\_p} = \{w_i \mid i \leq K, and \ \sum_{i=1}^{K} p_{w_i} \leq top\_p\} \quad (2)$$

(2) Temperature

Temperature adjusted the smoothing level of the model's output probability distribution. Given the original probabilities (p_w), the adjusted distribution was obtained by manipulating the temperature parameter (T) (usually a positive value) as below.

$$p\_w^{(T)} = \frac{\exp\left(\frac{\log p_w}{T}\right)}{\sum_{w'} \exp\left(\frac{\log p_{w'}}{T}\right)} \quad (3)$$

When T < 1, the distribution became more peaked, favoring the most probable words. When T > 1, the distribution became smoother, increasing the probability of less probable words.

(3) Frequency_penalty

The frequency_penalty was used to suppress the repeated occurrence of words, which worked by adding a penalty term to the model's logits, adjusting the probabilities of words to reduce the likelihood of words that had already appeared. Mathematically, assuming that in the current context, the word w had appeared count_w times, and the original logits of the model were l_w. With a penalty coefficient λ_{freq}, the adjusted logits were as below.

$$l'_w = l_w - \lambda_{freq} \times \text{count}_w \quad (4)$$

A negative λ_{freq} encouraged repetition (not commonly used). A positive λ_{freq} penalized words that had already appeared, reducing duplication.

(4) Presence_penalty

The presence_penalty was used to reduce repetition, but it operated based on whether a word had ever appeared rather than how many times. It typically involved adding a penalty term to the logits. Mathematically, presence_w was defined as a boolean variable indicating whether the word w had already appeared (1 or 0). With penalty coefficient λ_{pres}, the adjusted logits were

as follows.

$$l_{w'} = l_w - \lambda_{pres} \times \text{presence}_w \quad (5)$$

Parameter sensitivity algorithm

The algorithm employed a meticulously designed four-layer nested loop structure to systematically explore all possible combinations within the four key hyperparameters of the GPT-4 model, which included the top_p parameter ranging from 0.0 to 1.0 in increments of 0.2, the temperature parameter ranging from 0.0 to 2.0 in increments of 0.4, the presence_penalty parameter ranging from -2.0 to 2.0 in increments of 1.0, and the frequency_penalty parameter ranging from -2.0 to 2.0 in increments of 1.0. During the algorithm's initialization phase, the system configured the parameter step sizes and evaluation metric system and intelligently checked the status of the result storage file. If the file did not exist, it created a new Excel file containing all evaluation dimension headers, otherwise it automatically read the last row of parameters and resumed execution from the breakpoint. This design ensured both the continuity of experiments and the avoidance of resource waste caused by redundant runs. For each specific parameter combination, the algorithm first retrieved test questions from a predefined JSON-format question bank, then called the GPT-4 model *via* API to generate the corresponding textual responses. During application programming interface invocation, the algorithm employed an intelligent retry mechanism with exponential backoff. When encountering a server overload (HTTP 429 error), the system would automatically wait and retry. The waiting time for each retry increased incrementally (starting at 10 seconds and increasing by 5 seconds each time), up to a maximum of three retries. This design significantly enhanced robustness and completion rates in unstable network conditions. All generated answers were properly saved to intermediate files, preparing for subsequent batch evaluation. In the evaluation phase, the algorithm employed a comprehensive multi-dimensional and multi-metric assessment framework, which used BERTScore (https://vinija.ai/nlp/metrics/) to measure semantic similarity between generated and reference texts, obtaining three key metrics of precision, recall, F1 score, evaluate the fluency and naturalness of the texts using the BLEU algorithm with smoothing functions applied to mitigate biases caused by short texts, employ the Metric for Evaluation of Translation with Explicit Ordering (METEOR) algorithm to assess accuracy and readability, calculate surface feature similarity between texts using cosine weighted by Term Frequency-Inverse Document Frequency, compute perplexity using the GPT-2 model to evaluate the language model quality of the texts, and generate text fingerprints *via* the SimHash algorithm and compute Hamming distance to assess differences from an information theory perspective. This multi-angle evaluation system ensured the comprehensiveness and reliability of the experimental results. The algorithm automatically consolidated and recorded all parameter combinations and their corresponding evaluation results into an Excel file with each data row containing the complete parameter configuration and all evaluation metric results. The entire process automated parameter configuration, text generation, performance evaluation, and result recording, providing a systematic, efficient, and reliable experimental framework for LLM parameter optimization. This approach not only significantly improved the efficiency of parameter search but also offered rich data support and in-depth insights for model performance enhancement

Results and discussion

The factors affected precision

The performance of LLM across five core metrics including precision, recall, F1, cosine, and METEOR with fixed experimental parameters of 0.4 for top_p, 1.2 for temperature, and 0 for presence_penalty, while only frequency_penalty was set to -2.0, 0.0, and 2.0 to regulate lexical repetition demonstrated that, when frequency_penalty was -2.0 (encouraging word repetition), the model performed poorest across

Table 1. The impact of frequency_penalty on the results.

Frequency_penalty	Precision	Recall	F1	Cosine	METEOR
-2.0	0.2669	0.4158	0.3341	0.2143	0.1469
0.0	0.4573	0.6066	0.5282	0.5328	0.3374
2.0	0.4191	0.6050	0.5063	0.4994	0.3371

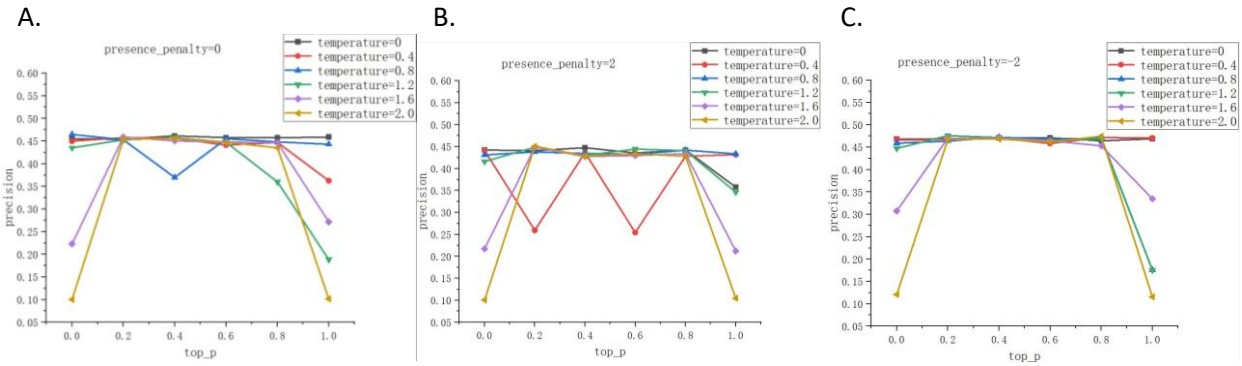

Figure 1. The relationship among the changes of presence_penalty, temperature, top_p, and precision.

all metrics with precision at 0.2669, F1 at 0.3341, and cosine at just 0.2143, indicating excessive repetition undermining output accuracy and semantic relevance. In contrast, frequency_penalty 0.0 (no lexical penalty) delivered the optimal overall performance with precision at the peak of 0.4573, recall at the highest 0.6066, F1 at a maximum 0.5282, and both cosine and METEOR reaching their top values of 0.5328 and 0.3374, respectively, proving a neutral stance on repetition balances diversity and accuracy. When frequency_penalty was 2.0 (suppressing repetition), the model's performance slightly declined compared to the 0.0 setting with precision dropping to 0.4191 and F1 to 0.5063, suggesting overly strict repetition suppression impaired semantic consistency. Therefore, frequency_penalty of 0.0 was the best choice under the given fixed parameters (Table 1). When presence_penalty was fixed at 0, most temperature groups maintained stable precision within the central top_p range (0.2 – 0.8). The temperature setting of 0 showed consistent precision across all top_p values, while higher temperature settings experienced sharp precision declines at extreme top_p values of 0.0 or 1.0 (Figure 1A). With presence_penalty setting to 2, the model's precision remained relatively stable for most temperature settings across most of the top_p range. However, the temperature setting of 0.4 exhibited notable volatility with precision dropping sharply at specific top_p values, while higher temperature settings still showed reduced precision at extreme top_p values (Figure 1B). When presence_penalty was set to -2, the model retained solid precision for moderate top_p values (0.2 – 0.8) across most temperature groups. The temperature setting of 0 maintained stable precision throughout the top_p range, while higher temperature settings experienced significant precision drops at extreme top_p values, and the temperature setting of 1.6 started with lower precision at top_p of 0.0 before recovering in the central range (Figure 1C). Overall, the results demonstrated that moderate top_p values of 0.2 – 0.8 consistently supported stable precision across all presence_penalty settings, while extreme top_p values paired with higher temperature settings reduced performance. Lower temperature settings enhanced precision stability, and neutral or slightly negative presence_penalty values yielded more consistent results compared to strict repetition suppression.

The factors affected recall

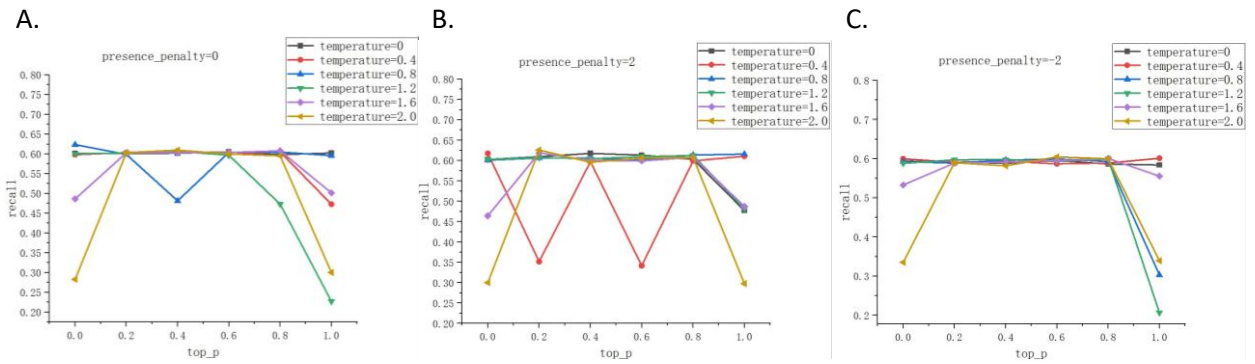

Figure 2. The relationship among the changes of presence_penalty, temperature, top_p, and recall.

When presence_penalty was fixed at 0, a neutral stance on topic repetition, most temperature groups maintained high recall within the central top_p range of 0.2 – 0.8. The temperature setting of 0 showed consistent recall across all top_p values, while higher temperature settings experienced sharp recall declines at extreme top_p values of 0.0 or 1.0 (Figure 2A). With presence_penalty was set to 2, which suppressed topic repetition, the model's recall remained relatively stable for most temperature settings across most of the top_p range. However, the temperature setting of 0.4 exhibited notable volatility with recall dropping sharply at specific top_p values, while higher temperature settings still showed reduced recall at extreme top_p values (Figure 2B). When presence_penalty was set to -2, which encouraged topic repetition, the model retained solid recall for moderate top_p values of 0.2 – 0.8 across most temperature groups. The temperature setting of 0 maintained stable recall throughout the top_p range, while higher temperature settings experienced significant recall drops at extreme top_p values, and the temperature setting of 1.6 started with lower recall at top_p of 0.0 before recovering in the central range (Figure 2C). The results demonstrated that moderate top_p values of 0.2 – 0.8 consistently supported stable recall across all presence_penalty settings, while extreme top_p values paired with higher temperature settings reduced performance. Lower temperature settings enhanced recall stability, and neutral or slightly negative presence_penalty values yielded more consistent results compared to strict repetition suppression.

The factors affected cosine

When presence_penalty was fixed at 0, most temperature groups maintained moderate to high cosine within the central top_p range of 0.2 – 0.8. The temperature setting of 0 showed consistent similarity across all top_p values, while higher temperature settings experienced sharp declines in similarity at extreme top_p values of 0.0 or 1.0 (Figure 3A). With presence_penalty being set to 2, the model's cosine remained relatively stable for most temperature settings across most of the top_p range. However, the temperature setting of 0.4 exhibited notable volatility with similarity dropping sharply at specific top_p values, while higher temperature settings still showed reduced similarity at extreme top_p values (Figure 3B). When presence_penalty was set to -2, the model retained solid cosine for moderate top_p values of 0.2 – 0.8 across most temperature groups. The temperature setting of 0 maintained stable similarity throughout the top_p range, while higher temperature settings experienced significant drops in similarity at extreme top_p values, and the temperature setting of 1.6 started with lower similarity at top_p of 0.0 before recovering in the central range (Figure 3C). The results demonstrated that moderate top_p values consistently supported stable semantic alignment across all presence_penalty settings, while extreme top_p values paired with

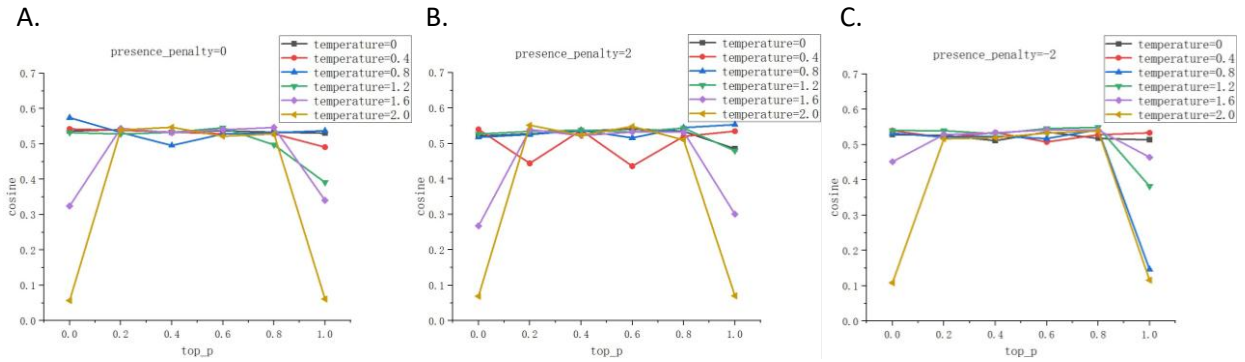

Figure 3. The relationship among the changes of presence_penalty, temperature, top_p, and cosine.

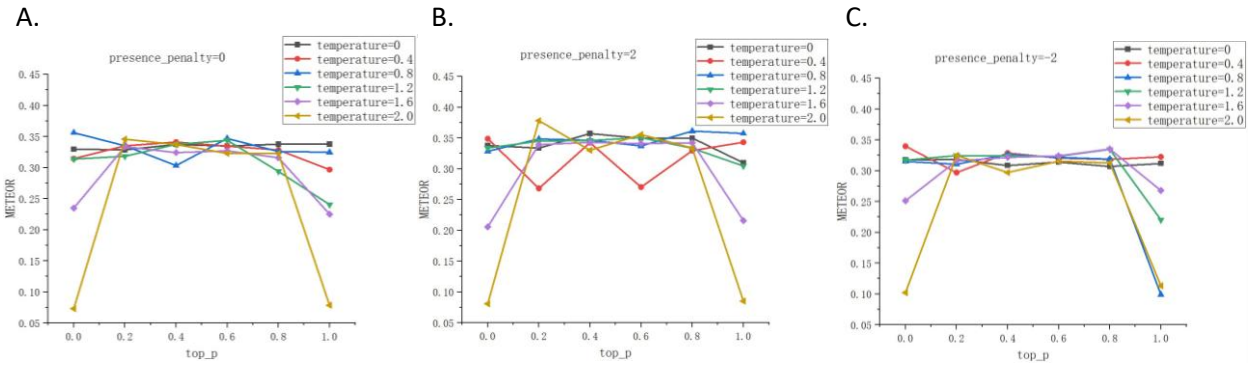

Figure 4. The relationship among the changes of presence_penalty, temperature, top_p, and METEOR.

higher temperature settings reduced performance. Lower temperature settings enhanced the stability of semantic alignment, and neutral or slightly negative presence_penalty values yielded more consistent results compared to strict repetition suppression.

The factors affected METEOR

When presence_penalty was fixed at 0, most temperature groups maintained moderate METEOR scores within the central top_p range. The temperature setting of 0 showed consistent scores across all top_p values, while higher temperature settings experienced sharp declines in scores at extreme top_p values (Figure 4A). With presence_penalty being set to 2, the model's METEOR score remained relatively stable for most temperature settings across the majority of the top_p range. However, the temperature setting of 0.4 exhibited notable volatility, with scores dropping sharply at specific top_p values, while higher temperature settings still showed reduced scores at extreme top_p values (Figure 4B). When presence_penalty was set to -2, the model retained solid METEOR scores for moderate top_p values across most temperature groups. The temperature setting of 0 maintained stable scores throughout the top_p range, while higher temperature settings experienced significant drops in scores at extreme top_p values, and the temperature setting of 1.6 started with lower scores at top_p of 0.0 before recovering in the central range (Figure 4C). The results indicated that moderate top_p values consistently supported stable text generation quality across all presence_penalty settings, while extreme top_p values paired with higher temperature settings reduced performance. Lower temperature settings enhanced the stability of text generation quality, and neutral or slightly negative presence_penalty values yielded more consistent results compared

to strict repetition suppression.

Conclusion

This research systematically explored the evaluation and optimization strategies of LLM for proteins with particular focus on the construction of test question banks and parameter tuning effects on model performance. Through automatic generation, quality control, and expert review, a high-quality, diverse set of protein-related test questions was developed to provide standardized testing data for model evaluation. The influences of various parameters including temperature, top_p, presence_penalty, and frequency_penalty on model outputs were analyzed to identify their roles in content diversity, coherence, and quality control. The results indicated that presence_penalty and frequency_penalty had a notable regulatory effect on model performance. Strictly positive presence_penalty values that suppressed topic repetition could lead to volatile performance for specific temperature settings, while excessively high values reduced content accuracy, recall, and semantic consistency. Moderate temperature settings of 0.8 – 1.2 balanced diversity and control effects, whereas values that were too high or too low were suboptimal. The impact of top_p was limited when maintained within a conventional range of 0.2 – 0.8, but extreme values of 0.0 or 1.0 caused significant performance reduction, especially when paired with higher temperature settings. Specifically, optimal performance was observed when penalty parameters were set to 0 or slightly negative, temperature was controlled within 0.8 – 1.2, and top_p was maintained at 0.2 – 0.8. This configuration enhanced generation quality, especially in high-precision or high-recall scenarios, by stabilizing precision, recall, semantic consistency, and text generation quality across all presence_penalty settings. This research not only provided a theoretical basis for improving the performance of protein-focused LLM but also filled a research gap in the construction of protein-specific test question banks, laying a solid foundation for future natural language processing applications in biomedical and related fields.

Acknowledgements

This research was supported by the 2024 Basic Scientific Research Business Expenses of Provincial Undergraduate Universities in Heilongjiang Province (Grant No. 2024-KYYWF-0902).

References

1. Fan W, Zhou Y, Wang S, Yan Y, Liu H, Zhao Q, *et al*. 2025. Computational protein science in the era of Large Language Models. Brief Bioinform. 26(2):bbae345.
2. Shen Y, Chen Z, Mamalakis M, He L, Xia H, Li T, *et al*. 2024. A fine-tuning dataset and benchmark for Large Language Models for protein understanding. In: 2024 IEEE Int Conf Bioinformatics Biomedicine (BIBM). IEEE. 2024:2390-2395.
3. Welleck S, Kulikov I, Roller S, Dinan E, Cho K, Weston J. 2020. Neural text generation with unlikelihood training. Proc 58th Annu Meet Assoc Comput Linguist. 2020:2876-2887.
4. Ravfogel S, Goldberg Y, Goldberger J. 2023. Conformal nucleus sampling. Findings Assoc Comput Linguist ACL 2023. 2023:27-34.
5. Mann B, Ryder N, Subbiah M, Kaplan J, Dhariwal P, Neelakantan A, *et al*. 2020. Language models are few-shot learners. Adv Neural Inf Process Syst. 33:1877-1901.
6. Lim ZW, Pushpanathan K, Yew SME, Lai Y, Sun CH, Lam JSH, *et al*. 2023. Benchmarking Large Language Models' performances for myopia care: A comparative analysis of ChatGPT-3.5, ChatGPT-4.0, and Google Bard. EBioMedicine. 95:104770.
7. Buscemi A. 2024. A comparative study of code generation using chatgpt 3.5 across 10 programming languages. J Syst Softw. 207:111892.
8. Wang C, Liu X, Awadallah AH. 2023. Cost-effective hyperparameter optimization for Large Language Model generation inference. In: Int Conf Autom Mach Learn. PMLR. 2023:1-17.
9. Arora C, Sayeed A, Licorish S, Wang F, Treude C. 2024. Optimizing Large Language Model hyperparameters for code generation. Empir Softw Eng. 29(4):89.
10. Arora C, Sayeed AI, Licorish S, Wang F, Treude C. 2024. Hyperparameter tuning for code generation in Large Language Models: A systematic evaluation. IEEE Trans Softw Eng. 50(5):1123-1137.
11. Jia M, Cui Z, Hug G. 2024. Enhancing LLMs for power system simulations: A feedback-driven multi-agent framework. IEEE Trans Smart Grid. 2024(16):5556-5572.

12. Shlomov S, Sela A, Levy I, Galanti L, Abitbol R. 2024. From grounding to planning: Benchmarking bottlenecks in web agents. Proc 2024 Conf Empir Methods Nat Lang Process. 2024:5201-5212.
13. Consortium U. 2015. UniProt: A hub for protein information. Nucleic Acids Res. 43(D1):D204-D212.
14. Burley SK, Berman HM, Kleywegt GJ, Markley JL, Nakamura H, Velankar S. 2017. Protein Data Bank (PDB): the single global macromolecular structure archive. Protein Crystallogr Methods Protocols. 2017:627-641.
15. Schoch CL, Ciufo S, Domrachev M, Hotton CL, Kannan S, Khovanskaya R, *et al*. 2020. NCBI taxonomy: A comprehensive update on curation, resources and tools. Database. 2020:baaa062.
16. Team K, Du A, Gao B, Xing B, Jiang C, Chen C, *et al*. 2025. Kimi k1.5: Scaling reinforcement learning with Large Language Models. Trans Mach Learn Res. 3:1-25.
17. Hesham A, Hamdy A. 2024. Fine-tuning GPT-4o-mini for programming questions generation. In: 2024 Int Conf Comput Appl (ICCA). IEEE. 2024:1-6.