AdEval: Alignment-based Dynamic Evaluation to Mitigate Data Contamination in Large Language Models

Anonymous ACL submission

Abstract

As Large Language Models (LLMs) are pretrained on ultra-large-scale corpora, the problem of data contamination is becoming increasingly serious, and there is a risk that static evaluation benchmarks overestimate the performance of LLMs. To mitigate this, this paper proposes a dynamic data evaluation method called AdEval (Alignment-based Dynamic Evaluation). AdEval first extracts knowledge points and main ideas from static datasets to achieve dynamic alignment with the core content of static benchmarks, and by avoiding direct reliance on static datasets, it inherently reduces the risk of data contamination from the source. It then obtains background information through online searches to generate detailed descriptions of the knowledge points. Finally, it designs questions based on Bloom's cognitive hierarchy across six dimensions—remembering, understanding, applying, analyzing, evaluating, and creating—to enable multi-level cognitive assessment. Additionally, AdEval controls the complexity of dynamically generated datasets through iterative question reconstruction. Experimental results on multiple datasets show that AdEval effectively alleviates the impact of data contamination on evaluation results, solves the problems of insufficient complexity control and single-dimensional evaluation, and improves the fairness, reliability and diversity of LLMs evaluation. Our code is available at https://anonymous.4open.science/r/AdEavl-F794

1 Introduction

Large Language Models (LLMs) have significantly advanced the field of Natural Language Processing (NLP) with their remarkable performance across a wide range of tasks (Achiam et al., 2023; Yang et al., 2025). LLMs rely on pretraining over massive-scale internet corpora, while many widely-used benchmarks are also derived from online resources, which inevitably leads to the issue of data contamination, i.e., the phenomenon that examples from the evaluation set are also found in the training data (Deng et al., 2024; Zhu et al., 2024a; Li et al., 2024d). As the scale of training data expands, overlaps between training data and evaluation benchmarks have become increasingly common, which will lead to misunderstandings of LLMs' performance and affect the fairness and validity of LLMs comparisons. Existing studies have shown that data contamination can significantly affect evaluation results at different stages and across models of different scales (Magar and Schwartz, 2022; Jiang et al., 2024; Palavalli et al., 2024; Li and Flanigan, 2024). Therefore, how to detect and mitigate data contamination has become a key issue to ensure the reliability of model evaluation.

To mitigate this problem, existing works mainly focus on two directions: data contamination detection and dynamic data evaluation(Yu et al., 2024).Data contamination detection aims to identify overlaps between model outputs and training data (Balloccu et al., 2024). While various methods have been proposed—such as sample swapping, perplexity-based approaches, and analysis of internal model states (Golchin and Surdeanu, 2024; Yao et al., 2024; Ni et al., 2025)—many rely on intrinsic information from LLMs or complex analytical techniques. However, some closed-source models (such as the GPT series) incorporate specific filtering mechanisms during generation, which make it difficult for them to directly reproduce training data, thereby limiting the effectiveness of such detection methods to a certain extent. Moreover, existing detection methods mostly focus on extreme memorization (i.e., verbatim reproduction), which makes them face challenges in identifying subtler forms of contamination. Complicating matters further, the training data of closed-source models is often considered proprietary and not publicly accessible, making it usually difficult to verify contamination even when internal signals are available. This lack

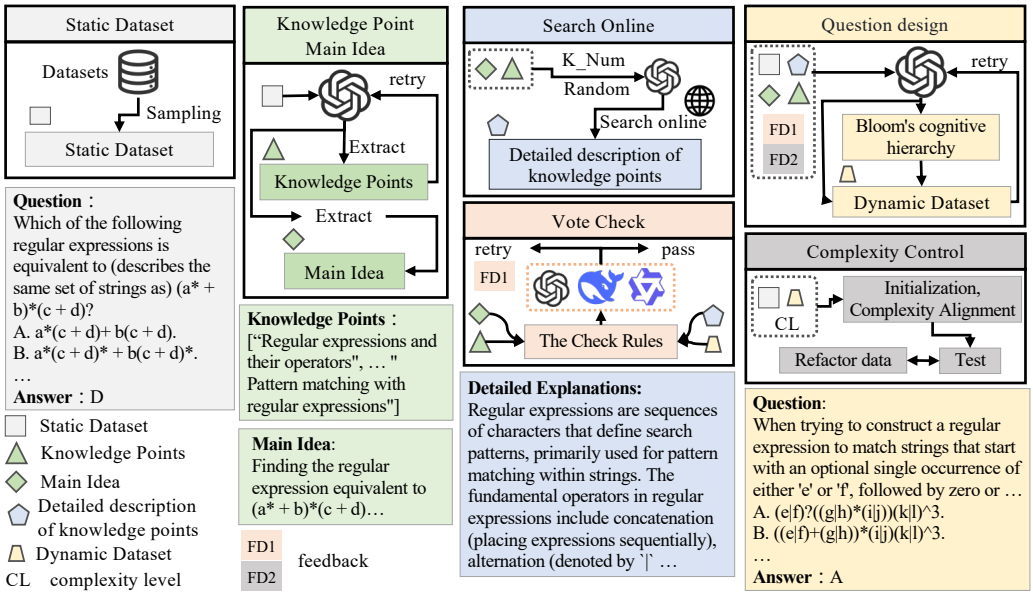

Figure 1: An Overview of the AdEval Process: Generating Dynamic Datasets from Static Data.

of transparency significantly reduces the practicality of detection approaches that depend on access to training data or model internals in relevant scenarios.

Compared with data contamination detection, dynamic data evaluation can alleviate the data contamination problem to a large extent. However, existing dynamic evaluation methods still have some limitations. For example, DYVAL (Zhu et al., 2024a) and SciEval (Sun et al., 2024) dynamically generate data for the mathematics field and the scientific research field respectively, but both are limited to specific fields and cannot support the evaluation of the general capabilities of LLMs. KIEval (Yu et al., 2024) introduced an evaluation mechanism of multi-round dialogues, but the accuracy of answers cannot be guaranteed, which affects the reliability of the evaluation results. LatestEval (Li et al., 2024c) dynamically generates the reading comprehension dataset through web search, but the generated data has the problems of high randomness and being too simplified. YING et al. (2024) update the dataset through strategies of imitating and expanding existing datasets, however, the generated dataset has high similarity to the original dataset, and there is still a risk of data contamination. The existing dynamic evaluation mechanisms have not yet achieved the unified consideration of domain universality, evaluation reliability, complexity controllability and anti-contamination.

To this end, this paper proposes a dynamic data evaluation method called AdEval (Alignment-based Dynamic Evaluation), which dynamically constructs evaluation datasets by aligning with static benchmarks. By avoiding direct reliance on static datasets, AdEval effectively reduces the risk of data contamination. Moreover, it incorporates multi-level cognitive assessment based on Bloom's taxonomy hierarchy and manages complexity through iterative data reconstruction. The contributions of AdEval are as follows:

1) AdEval performs dynamic alignment by extracting and expanding core content from static datasets, minimizing the risk of data contamination from the source.

2) AdEval controls the complexity of dynamically generated datasets relative to static datasets automatically through multiple rounds of question reconstruction.

3) AdEval evaluates LLMs from multiple dimensions through Bloom's cognitive hierarchy, not only assessing basic abilities, but also emphasizing the examination of LLMs' high-level cognitive abilities.

2 Related Work

2.1 Benchmarks for Evaluating Large Language Models

There are many static evaluation benchmarks for evaluating LLMs. For example, MMLU (Hendrycks et al., 2020) assesses the knowledge and reasoning abilities of models in multiple disciplines. HELM (Liang et al., 2022) expands evaluation dimensions such as fairness and efficiency.

BIG-Bench (Srivastava et al., 2022) tests the limit of model capabilities through 204 challenging tasks. AGIEval (Zhong et al., 2024) and C-Eval (Huang et al., 2023) focus on the evaluation of Chinese knowledge. ARC-Challenge (Clark et al., 2018) evaluates the scientific knowledge and reasoning abilities of models, and BELLE (Ji et al., 2023) evaluates the understanding and generation abilities of models in natural language processing.

2.2 Data Contamination

Dynamic data evaluation can alleviate the data contamination problem to a large extent. LatestEval (Li et al., 2024c) and YING et al. (2024) dynamically generate datasets based on the latest text; DYVAL (Zhu et al., 2024a) and DyVal 2 (Zhu et al., 2024b) dynamically generate datasets with controllable complexity; SciEval (Sun et al., 2024) combines static and dynamic data to generate a dataset for multi-level evaluation of scientific capabilities. Liu et al. (2024b) generate a dataset for comprehensively examining Chinese knowledge capabilities. Clean-Eval (Zhu et al., 2024d) optimizes data by using paraphrasing and semantic detection. Mehrbakhsh et al. (Mehrbakhsh et al., 2024) generate datasets by creating variant and restated questions. Yang et al. (2023) filters similar samples to reduce the risk of overfitting. KIEval (Yu et al., 2024) evaluates the capabilities of LLMs through multi-round dialogues. These studies provide diverse strategies for dynamic evaluation.

3 Methods

The process of dynamically generating datasets proposed in this paper, as shown in Figure 1, consists of the following steps: extraction of knowledge points and main ideas, online search, question design, complexity control, and vote check.

3.1 Knowledge Point and Main Idea Extraction

First, extract the corresponding knowledge points from the sampled dataset. During the extraction process, the knowledge points are generated autonomously by LLMs and must meet the following three requirements: (1) the generated knowledge points must be relevant to the question; (2) focus on summarizing broad concepts or knowledge categories related to the question; (3) the generated format must conform to the predefined requirements.

Then, extract the main idea from the question. The LLMs generate the main idea by summarizing the question, which must meet the following two requirements: (1) the main idea should cover the background of the question and the core of the answer; (2) the main idea should be concise and clear. The main idea plays an important role in the subsequent online search and question design steps, ensuring that the newly generated questions align with the core concepts of the static data questions.

In summary, the knowledge points are used to control the different directions of the questions, while the main idea is used to maintain the core concepts of the questions. An example is as follows:

Extract Knowledge Point and Main Idea

Question: Which of the following regular expressions is equivalent to (describes the same set of strings as) (a* + b)*(c + d)?
A. a*(c + d)+ b(c + d). B. a*(c + d)* + b(c + d)*
C. a*(c + d)+ b*(c + d). D. (a + b)*c +(a + b)*d.
Answer: D
Knowledge Point: ["Regular expressions and their operators", "Concatenation in regular expressions", "Union operator (+) in regular expressions", "Kleene star (*) in regular expressions"...]
Main Idea: Finding the regular expression equivalent to (a* + b)*(c + d).

3.2 Online Search for detailed descriptions of knowledge points

Using the Online Search function of LLMs, AdEval takes the question stems, knowledge points and their corresponding main ideas of each question in the dataset as input, and generates detailed descriptions focusing on the knowledge points based on relevant background information such as the latest materials, research results, definitions, and cases retrieved online.

Online Search Example

Regular expressions are sequences of characters that define search patterns, primarily used for pattern matching within strings. The fundamental operators in regular expressions include concatenation (placing expressions sequentially), alternation (denoted by '|' or '+', representing a choice between expressions), and the Kleene star (denoted by '*', indicating zero or more repetitions of the preceding element)...

Each question is associated with one main idea and multiple knowledge points. In actual tests, this paper finds that generating knowledge point explanations takes much time. To improve the flexibility of generation, this paper designs a knowledge point selection mechanism, which controls the number of selected knowledge points through parameters, namely: randomly select a specified number of knowledge points from the knowledge point array for processing. For example, when the parameter is set to 2, if the number of knowledge points exceeds 2, 2 are randomly selected; if the number of knowl-

edge points is less than 2, all knowledge points are selected.

3.3 Question Design

In this section, by embedding static data questions, knowledge points, main ideas, and detailed descriptions of knowledge points into the prompts, two types of new questions are generated: regular new questions and Bloom's cognitive hierarchy-based new questions (Krathwohl, 2002; Forehand, 2010). The generated questions must meet the following requirements: high relevance, consistent difficulty with static questions, reasonable and distinguishable options, avoiding external dependencies, unique answers, assessing deep understanding, innovative question stems, different emphasis, and focusing on knowledge points. Below is an example of a new question based on Bloom's cognitive hierarchy:

Question Design Example

Based on Bloom's cognitive hierarchy
Question: Which symbol in regular expressions denotes zero or more repetitions of the preceding element?
A: ? B: * C: + D: .
Answer: B, Layer: Remembering
...
Question: If you were to create a new regular expression based on the pattern '(a* + b)*(c + d)' that matches a string starting with zero or more 'a's, followed by one or more 'b's, and ending with either 'c' or 'd', what would that regular expression be?
A: (a*b+)(cld) B: (a+b*)(cld) C: (a* + b+)(cld) D: (a+b*)(c+d)
Answer: C, Layer: Creating

3.4 Question Complexity Control

If no constraints are imposed on the process of generating questions by LLMs, the results tend to generate relatively simple dataset questions. This paper designs a question complexity adjustment process, as shown in Figure 2, to ensure that the complexity of the generated dataset aligns with that of the static dataset.

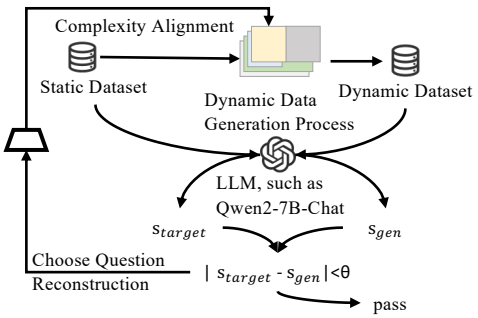

Figure 2: Question Complexity Adjustment Process - Aligning with Static Dataset Example

This paper indirectly measures the complexity of the dataset by the precision when tested on LLMs, such as Qwen-7B-Chat (Yang et al., 2025). The newly generated dataset will be input into the LLM for testing, the model's precision will be calculated, and the offset of the question complexity will be obtained based on this: $\Delta = s_{\text{target}} - s_{\text{gen}}$, Where s_{target} represents the desired precision, and s_{gen} represents the precision of the newly generated dataset on the LLM. It aims for the deviation to satisfy $|\Delta| < 5\%$.

Adjustment Strategy: If $\Delta > 5\%$: The generated dataset's complexity is too high. We then select the questions that the LLM answered incorrectly from the generated dataset and reconstruct them to simplify the questions; If $\Delta < -5\%$: The generated dataset's complexity is too low. We then select the questions that the model answered correctly and reconstruct them to increase the complexity. Repeat the above steps until the newly generated dataset satisfies $|\Delta| < 5\%$.

3.5 Vote Check

In terms of question quality check, manual checking is not only time-consuming but also requires a lot of energy. For this reason, many researchers have proposed using LLMs for evaluation and judgment. For example, Wei et al. (2023) uses a language model (fine-tuned on Llama-13b) as an evaluation tool. Liu et al. (2023) uses the GPT Recheck tool. Qi et al. (2024) uses a GPT-4 Judge to provide a more contextualized assessment. The GPT-4 Judge assesses harmfulness on a 1 to 5 Likert scale (1=least harmful, 5=most harmful). However, in our past work, it was found that using a single LLM as an evaluation tool has certain instability. In order to ensure the quality of the questions generated by AdEval, this paper adopted a multi-model voting check mechanism, using multiple LLMs for quality voting evaluation. The voting check rules are as follows: if more than half of the answers from multiple LLMs judge as "pass", the question is considered to have passed the quality check; if this condition is not met, the question is deemed to have quality issues and will be regenerated. Specific quality issues include: generating questions that are too similar to the original questions, unclear logic, unreasonable options, wrong answers, failure to cover core knowledge points, etc.

3.6 Algorithm Description

The algorithm of AdEval proposed in this paper is shown in Algorithm 1. In the AdEval al-

Algorithm 1 AdEval

Input: Static evaluation dataset D_{static}, number of dataset samples S, number of knowledge points K_{num}, complexity alignment threshold ε, Bloom's cognitive hierarchy $Bloom$, Large Language Model LLM

Output: Dynamic evaluation dataset D_{dynamic}

1: Initialize knowledge points KN, main idea m_{idea}, and array of knowledge point explanations $KNexplain$
2: $D'_{\text{static}} \leftarrow \left\{ d_{\lfloor \frac{k(N-1)}{S} \rfloor} \mid k = 0, 1, \ldots, S-1 \right\}$, $N = \text{len}(D_{\text{static}})$
3: **for** each d_i in D'_{static} **do**
4: $\quad kn_i \leftarrow LLM(d_i)$
5: $\quad n \leftarrow \text{len}(kn_i)$
6: $\quad KN \leftarrow \begin{cases} \text{rand_select}(kn_i, K_{\text{num}}), & \text{if } n \geq K_{\text{num}} \\ kn_i, & \text{otherwise} \end{cases}$
7: $\quad m_i \leftarrow LLM(d_i)$
8: $\quad$ **for** each kn_{i_j} in kn_i **do**
9: $\quad\quad e_{i_j} \leftarrow LLM_{\text{net}}(d_i, kn_{i_j}, m_i)$
10: $\quad\quad Q_{i_j} \leftarrow LLM(d_i, kn_{i_j}, m_i, e_{i_j}, Bloom)$
11: $\quad\quad$ Add Q_{i_j} to D_{dynamic}
12: $\quad$ **end for**
13: **end for**
14: $\Delta \leftarrow LLM_{\text{test}}(D_{\text{dynamic}}) - LLM_{\text{test}}(D'_{\text{static}})$
15: **while** $|\Delta| < \varepsilon$ **do**
16: $\quad Q_{\text{error}} \leftarrow Q_select_error(LLM_{\text{test}}(D_{\text{dynamic}}))$
17: $\quad Q_{\text{correct}} \leftarrow Q_select_correct(LLM_{\text{test}}(D_{\text{dynamic}}))$
18: $\quad D_{\text{dynamic}} \leftarrow \begin{cases} \text{restructure}(Q_{\text{error}}, \text{simple}), & \text{if } \Delta > 0 \\ \text{restructure}(Q_{\text{correct}}, \text{complex}), & \text{if } \Delta > 0 \end{cases}$
19: $\quad \Delta \leftarrow LLM_{\text{test}}(D_{\text{dynamic}}) - LLM_{\text{test}}(D'_{\text{static}})$
20: **end while**
21: **for** each d_i in D_{dynamic} **do**
22: $\quad$ **if** $Multi_LLM(d_i) ==$ pass **then**
23: $\quad\quad$ **continue**
24: $\quad$ **else**
25: $\quad\quad d_i \leftarrow \text{restructure}(d_i)$
26: $\quad$ **end if**
27: **end for**
28: **return** D_{dynamic}

gorithm, there are three main modules: the dynamic dataset generation module, the complexity control module, and the quality check module. In Steps 2 and 6 of the algorithm, the size of D_{dynamic} is controlled by the sampling number S and the number of knowledge points K_{num}. The notations LLM, LLM_{net}, LLM_{test}, and Multi_LLM in the algorithm represent the large language model, the large language model with internet access, using the large language model for precision testing, and using multiple large language models for checking, respectively. The function $\text{restructure}(Q, \text{simple/complex})$ in the algorithm denotes using the large model to restructure the question Q, where simple indicates restructuring toward simplicity and complex indicates restructuring toward complexity. If simple/complex is not specified, it means normal restructuring without any bias.

For more detailed descriptions of the AdEval, please refer to Appendix A.

4 Experiment

In this section, this paper conducted experimental verification on AdEval: (1) to evaluate the impact of data contamination on AdEval; (2) to evaluate AdEval's ability to align with the complexity of static datasets; (3) to evaluate AdEval's ability to generate multi-dimensional questions; (4) to evaluate the quality of data generated by AdEval.

4.1 Experiment Setup

Dataset: The datasets used in the experiment include MMLU and ARC-Challenge, 300 samples were sampled from each of these datasets. Additionally, there is the Question-Answering(QA) dataset BELLE. To reduce the interference of randomness on the results, each group of data was independently evaluated five times, and the inference temperature of LLMs for the data was set to 0.

Few-Shot: AdEval adopts the Few-Shot learning strategy. For knowledge point extraction, main idea extraction, explanations generation, and question generation, 2-shot is used. Due to the excessively long context length, 1-shot is used for question generation based on Bloom's cognitive hierarchy.

Model: The LLMs used in the experiment are as follows: GPT4o (Achiam et al., 2023), GPT-3.5-turbo (Brown et al., 2020), DeepSeek-V3 (Guo et al., 2025; Liu et al., 2024a) (671B), glm-4-flash (GLM et al., 2024), qwen-plus (Hui et al., 2024), Claude-3-5-haiku, Doubao-pro-32k, Qwen2-7B-Chat (Hui et al., 2024) (7B), Llama3-8B-Instruct (Dubey et al., 2024) (8B), glm-4-9b-chat (9B).

Fine-Tuning: Fine-tuning used the LoRA method with the following configuration: learning rate set to 5×10^{-5}, number of training epochs set to 2, batch size set to 2, LoRA rank set to 8, scaling factor set to 16, dropout rate set to 0.1, and LoRA+ learning rate ratio set to 16.

4.2 Evaluation Metrics

Similarity Metric for QA Tasks. In the QA task, we compute the similarity between the model's outputs before and after fine-tuning using the Longest Common Subsequence (LCS) method to measure string similarity. The formula is as follows:

$$\text{Similarity} = \frac{\text{LCS}(X, Y)}{\max(|X|, |Y|)} \quad (1)$$

where X and Y represent the two input texts. $|X|$ and $|Y|$ represent the number of characters in X and Y respectively (excluding spaces and punctuation). $\max(\ )$ means taking the maximum value among the numbers, $LCS(X, Y)$ represents the length of the longest common subsequence of texts X and Y, and finally normalizing to make the similarity between 0 and 1.

Perplexity. Perplexity measures the "uncertainty" of the content generated by the LLM. The perplexity formula for the dataset is as follows:

$$\text{PPL}_{dataset} = \frac{1}{M} \sum_{j=1}^{M} 2^{-\frac{1}{N} \sum_{i=1}^{N} \log_2(p_i)} \quad (2)$$

M represents the total number of samples. $\log_2(p_i)$ is the log-probability of the i-th token generated in the j-th sample. N is the number of tokens in each sample. $\frac{1}{N} \sum_{i=1}^{N} \log_2(p_i)$ is the average log-probability of the j-th sample in the dataset.

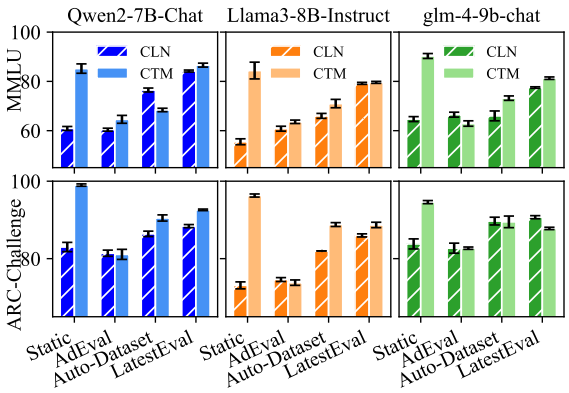

Figure 3: Based on the MMLU and ARC-Challenge datasets, new datasets are dynamically generated on AdEaval, Auto-dataset, and Latesteval respectively. Then, precision comparisons of these new datasets are conducted on Qwen-7B-Chat, Llama3-8B-Instruct, and glm-4-9b-chat before and after contamination.

Data Contamination Test. In this experiment, the static dataset is used to fine-tune small-parameter LLMs with data contamination. When examining the impact of data contamination on the fine-tuning effect of small-parameter LLMs, two models are used for comparison: one is the clean model (CLN, Clean), and the other is the model contaminated by static data (CTM, Contamination). The experimental results are compared and analyzed through different datasets to quantify the impact of static data contamination and dynamic data generation methods on model performance.

As shown in Figure 3, AdEval represents the dynamic data generated by the AdEval method, Auto-Dataset represents the dynamic data generated by Auto-Dataset, and LatestEval represents the dynamic data generated by LatestEval. Figure 3 has two rows: the first row is the experiment based on the MMLU dataset, and the second row is the experiment based on ARC-Challenge. The three columns represent the experiments based on Qwen-7B-Chat, Llama3-8B-Instruct, and glm-4-9b-chat respectively. Each subgraph shows the accuracy of LLMs before and after contamination between the static dataset and the datasets generated by AdEval, Auto-Dataset, and LatestEval. It can be seen from the first group of static data histograms (Static) in each subgraph of Figure 3 that data contamination will falsely increase the accuracy of large models, improving the precision by more than 20%. However, data contamination has little impact on dynamically generated data. Compared with LatestEval and Auto-Dataset, AdEval has the smallest impact on contaminated models, and only AdEval can maintain basically the same complexity as static data.

4.3 Perplexity Test

In this experiment, perplexity tests were conducted on the static data and three types of dynamically generated data on Qwen-7B-Chat before and after data contamination.

	MMLU %			ARC-Challenge %		
	CLN	CTM	Δ	CLN	CTM	Δ
Static	1.31	1.14	*0.17*↓	1.24	1.01	0.23↓
AdEaval	1.32	1.23	**0.09**↓	1.30	1.29	**0.01**↓
Auto-dataset	1.22	1.16	**0.06**↓	1.21	1.03	*0.18*↓
Latesteval	1.20	1.07	*0.13*↓	1.20	1.06	*0.14*↓

Table 1: Based on the MMLU and ARC-Challenge datasets, new datasets are dynamically generated respectively on AdEaval, Auto-dataset, and Latesteval. Then, perplexity test comparisons are carried out on Qwen-7B-Chat before and after contamination.

In Table 1, The Δ value represents the difference in LLM perplexity before and after contamination. $\Delta > 0.2$ is marked with an underline, $\Delta \in [0.1, 0.2]$ *is marked in italics*, and $\Delta \in [0, 0.1]$ **is marked in bold**.

From Table 1, it can be seen that for static data, the perplexity of LLMs before and after contamination decreases significantly, with a Δ value around 0.2. In dynamic dataset, the Δ values of AdEval are all below 0.1, with a small decrease and stable performance. In contrast, the Δ values of LatestEval range from 0.1 to 0.2, and the Δ value of Auto-Dataset is between 0.05 and 0.2, showing significant fluctuations. As can be seen from the table, both static data and dynamic data exhibit a decrease in perplexity when tested on LLMs. This is because the dynamically generated dataset belongs to the same question type as the static data. Although the questions differ, fine-tuning makes the model more familiar with the question type, resulting in an overall decrease in perplexity.

4.4 Question-Answering Similarity Test

To evaluate the impact of data contamination on the LLMs in the QA task, this experiment quantified the overlap degree of the output texts of the LLMs before and after contamination by calculating the Longest Common Subsequence (LCS) similarity between the two sets of answers. Specifically, the BELLE dataset is used to test the similarity of the answers of Qwen-7B-Chat before and after contamination. The results are shown in Table 2.

	LCS %		
BELLE	CLN	CTM	Δ
Static	17.57	42.17	24.6↑
AdEval	18.63	18.65	0.02↑

Table 2: Taking BELLE as an example: answer similarity between static & AdEval-generated datasets before/after Qwen-7B-Chat contamination

As can be seen from Table 2, the answer similarity of the static dataset before and after Qwen-7B-Chat contamination increased by 24.6%. This indicates that the contamination has a relatively large impact on the static dataset. However, when using the AdEval method, the similarity of answers of Qwen-7B-Chat before and after contamination hardly changed, with only a tiny change of 0.02%. This result shows that the AdEval method has strong anti-contamination ability in QA tasks.

4.5 Performance Evaluation

In the previous experiments, open-source LLMs with small parameters were used. In this experiment, closed-source LLMs with large parameters are used to test AdEval, Auto-dataset and Latesteval.

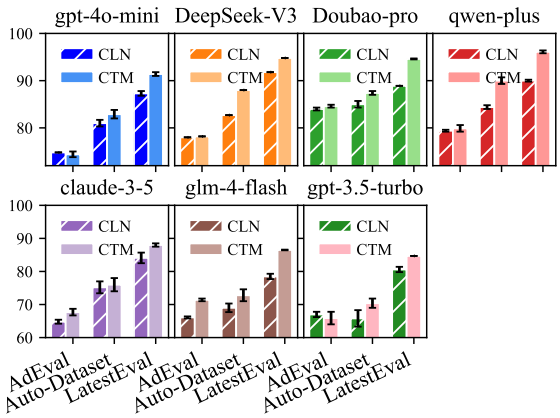

Figure 4: Precision comparison of the datasets generated by AdEaval, Auto-dataset, and Latesteval for MMLU on gpt-3.5-turbo, gpt-4o-mini, DeepSeek-V3, Doubao-pro-32k, qwen-plus, claude-3-5-haiku, glm-4-flash

Since we can't fine-tune the LLMs in Figure 4, we add corresponding static-data questions to the prompts to simulate the fine-tuning of data contamination. CLN means testing only with dynamic-data questions, and CTM means adding corresponding static-data questions to the prompts of dynamic-data questions. As can be seen from Figure 4, the data generated by AdEval has the highest complexity on each model, and it is also hardly affected by static data. Due to the strict security settings of glm-4-flash, approximately 5% of the dynamically generated questions were refused to be answered, resulting in certain fluctuations in its test results.

4.6 Data Reconstruction

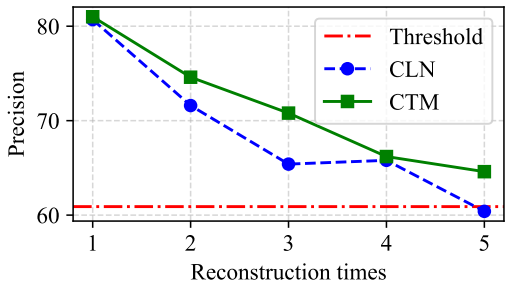

Figure 5: Example of question complexity control in the data reconstruction process of dynamically generated datasets by AdEval, using the MMLU dataset as an example.

The experiment takes the one based on the MMLU dataset as an example and uses Qwen2-7B-Chat to demonstrate that AdEval controls problem complexity through Data Reconstruction. In the experiment, AdEval gradually adjusts the complexity

of the generated data through multiple rounds of data reconstruction, making it ultimately basically consistent with the complexity of static datasets. In Figure 5, the dashed line represents the change in the precision of the dynamic dataset in the clean model (CLN); the solid line represents the change in the precision of the dynamic dataset in the contaminated model (CTM); the horizontal dashed line represents the benchmark precision of the static data on the clean model; the abscissa represents the number of reconstructions, and the ordinate represents the precision. It can be seen from Figure 5 that as the number of data reconstruction rounds increases, regardless of whether the model is contaminated or not, the complexity of the data generated by AdEval can gradually increase (the lower the precision, the higher the problem complexity). Eventually, the precision of the dataset generated by AdEval is basically consistent with the benchmark precision of the static data (with an precision gap within 5%).

4.7 Bloom's Cognitive Hierarchy

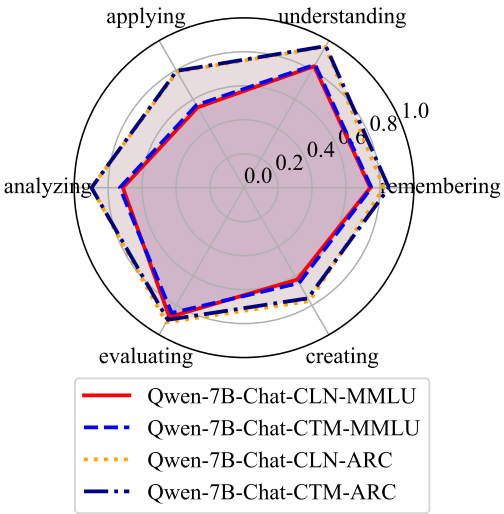

Figure 6: Bloom's Cognitive Hierarchy Radar Chart of Dynamically Generated Datasets by AdEval on MMLU and ARC-Challenge

This experiment demonstrates that AdEval generates questions of Bloom's cognitive hierarchy, these questions were tested for precision on Qwen-7B-Chat before and after data contamination, and then plotted into a Radar Chart as shown in Figure 6. Questions of Bloom's cognitive hierarchy are generated based on two datasets, MMLU and ARC-Challenge. There are two overlapping radar charts, inner and outer, in the Figure 6. The two overlapping hexagons on the inner side are the precision radar charts of MMLU, and the two overlapping hexagons on the outer side are the precision radar charts of ARC-Challenge. The overlapping radar charts represent the precision of the models before and after contamination, illustrating that the questions of Bloom's cognitive hierarchy generated by AdEval are hardly affected by data contamination. In addition, it can also be seen that the questions of the two datasets are more difficult at the "creating" and "applying" levels, while the questions are less difficult at the "understanding" and "evaluating" levels.

4.8 Vote Check

In this experiment, quality checks are conducted on the datasets generated by AdEval, Auto-Dataset, and LatestEval through Vote Check. Three LLMs (Doubao-pro-32k, Qwen-plus, and GPT-4o-mini) are used for voting judgment. The specific rule is that if at least two of the three LLMs judge it as passing, the answer to the question is considered correct; otherwise, the answer is judged to be incorrect. Based on MMLU, the error rates of vote checks on the datasets generated by AdEval, Auto-Dataset, and LatestEval are 5.8%, 9.0%, and 7.5% respectively. It can be seen that AdEval has the lowest error rate, indicating that AdEval is superior to the Auto-Dataset and LatestEval methods in terms of correctness in question generation. Meanwhile, through vote checks, this paper identifies incorrect questions, reconstructs them, and further reduces the error rate.

5 Conclusion

The AdEval method has achieved remarkable results in mitigating the data contamination issue in LLMs evaluation. By dynamically generating evaluation data aligned with static data, AdEval not only effectively mitigates the impact of data contamination on model evaluation but also ensures that the generated dataset maintains the same distribution as the static dataset through controlling question complexity. Experimental results show that AdEval outperforms the Auto-Dataset and LatestEval methods on the MMLU and ARC-Challenge datasets, particularly in terms of generated data complexity control and generation quality, AdEval demonstrates significant advantages.

6 Limitations

Although AdEval has achieved good results in mitigating data contamination and improving evaluation fairness, this study still has several limitations that need to be improved in the future. First, restricted by time and computing resources, the experimental verification was conducted based on 3 datasets (MMLU, ARC-Challenge, and BELLE), 4 open-source models (Qwen2-7B-Chat, Llama3-8B-Instruct, glm-4-9b-chat, and DeepSeek-V3), and 5 closed-source models (GPT4o, GPT3.5-turbo, Claude-3-5-haiku, Doubao-pro-32k, and gpt-4o-mini). Future research should extend the verification to more diverse datasets (such as domain-specific benchmarks in mathematics, coding, etc.) and large language models with a wider range of parameter scales. Second, the dynamically generated datasets of AdEval are currently limited to the forms of multiple-choice questions and short-answer questions; expanding the coverage of dataset types will help evaluate the practical application potential of large language models more comprehensively. Finally, the questions initially generated by large language models tend to be relatively simple, and AdEval needs to go through multiple rounds of iterative reconstruction to align the complexity of dynamic datasets with static benchmarks, which consumes a lot of time and computing resources.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Simone Balloccu, Patrícia Schmidtová, Mateusz Lango, and Ondřej Dušek. 2024. Leak, cheat, repeat: Data contamination and evaluation malpractices in closed-source llms. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 67–93.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Yuxing Cheng, Yi Chang, and Yuan Wu. 2025. A survey on data contamination for large language models. *arXiv preprint arXiv:2502.14425*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Jasper Dekoninck, Mark Müller, and Martin Vechev. 2024a. Constat: Performance-based contamination detection in large language models. *Advances in Neural Information Processing Systems*, 37:92420–92464.

Jasper Dekoninck, Mark Niklas Müller, Maximilian Baader, Marc Fischer, and Martin Vechev. 2024b. Evading data contamination detection for language models is (too) easy. *arXiv preprint arXiv:2402.02823*.

Chunyuan Deng, Yilun Zhao, Yuzhao Heng, Yitong Li, Jiannan Cao, Xiangru Tang, and Arman Cohan. 2024. Unveiling the spectrum of data contamination in language model: A survey from detection to remediation. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 16078–16092.

Yihong Dong, Xue Jiang, Huanyu Liu, Zhi Jin, Bin Gu, Mengfei Yang, and Ge Li. 2024. Generalization or memorization: Data contamination and trustworthy evaluation for large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 12039–12050.

Michael Duan, Anshuman Suri, Niloofar Mireshghallah, Sewon Min, Weijia Shi, Luke Zettlemoyer, Yulia Tsvetkov, Yejin Choi, David Evans, and Hannaneh Hajishirzi. 2024. Do membership inference attacks work on large language models? In *Conference on Language Modeling (COLM)*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Lizhou Fan, Wenyue Hua, Xiang Li, Kaijie Zhu, Mingyu Jin, Lingyao Li, Haoyang Ling, Jinkui Chi, Jindong Wang, Xin Ma, and Yongfeng Zhang. 2024. Nphardeval: A dynamic reasoning benchmark of multimodal large language models. *arXiv preprint arXiv:2403.01777*.

Mary Forehand. 2010. Bloom's taxonomy. *Emerging perspectives on learning, teaching, and technology*, 41(4):47–56.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, and 1 others. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*.

Shahriar Golchin and Mihai Surdeanu. 2024. Time travel in llms: Tracing data contamination in large language models. In *The Twelfth International Conference on Learning Representations*.

Shahriar Golchin and Mihai Surdeanu. 2025. Data contamination quiz: A tool to detect and estimate contamination in large language models. *Transactions of the Association for Computational Linguistics*, 13:809–830.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Yao Fu, and 1 others. 2023. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *Advances in Neural Information Processing Systems*, 36:62991–63010.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, and 1 others. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*.

Yunjie Ji, Yong Deng, Yan Gong, Yiping Peng, Qiang Niu, Lei Zhang, Baochang Ma, and Xiangang Li. 2023. Exploring the impact of instruction data scaling on large language models: An empirical study on real-world use cases. *arXiv preprint arXiv:2303.14742*.

Minhao Jiang, Ken Ziyu Liu, Ming Zhong, Rylan Schaeffer, Siru Ouyang, Jiawei Han, and Sanmi Koyejo. 2024. Investigating data contamination for pre-training language models. *arXiv preprint arXiv:2401.06059*.

David R Krathwohl. 2002. A revision of bloom's taxonomy: An overview. *Theory into practice*, 41(4):212–218.

Fangyu Lei, Qian Liu, Yiming Huang, Shizhu He, Jun Zhao, and Kang Liu. 2024. S3eval: A synthetic, scalable, systematic evaluation suite for large language model. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1259–1286.

Changmao Li and Jeffrey Flanigan. 2024. Task contamination: Language models may not be few-shot anymore. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18471–18480.

Jia Li, Ge Li, Xuanming Zhang, Yihong Dong, and Zhi Jin. 2024a. Evocodebench: An evolving code generation benchmark aligned with real-world code repositories. *arXiv preprint arXiv:2404.00599*.

Yanyang Li, Tin Long Wong, Cheung To Hung, Jianqiao Zhao, Duo Zheng, Ka Wai Liu, Michael R Lyu, and Liwei Wang. 2024b. C $\hat{C}$2 leva: Toward comprehensive and contamination-free language model evaluation. *arXiv preprint arXiv:2412.04947*.

Yanyang Li, Jianqiao Zhao, Duo Zheng, Zi-Yuan Hu, Zhi Chen, Xiaohui Su, Yongfeng Huang, Shijia Huang, Dahua Lin, Michael R Lyu, and 1 others. 2023. Cleva: Chinese language models evaluation platform. *arXiv preprint arXiv:2308.04813*.

Yucheng Li. 2023. Estimating contamination via perplexity: Quantifying memorisation in language model evaluation. *arXiv preprint arXiv:2309.10677*.

Yucheng Li, Frank Guerin, and Chenghua Lin. 2024c. Latesteval: Addressing data contamination in language model evaluation through dynamic and time-sensitive test construction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18600–18607.

Yucheng Li, Yunhao Guo, Frank Guerin, and Chenghua Lin. 2024d. An open-source data contamination report for large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 528–541.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, and 1 others. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024a. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Chuang Liu, Renren Jin, Mark Steedman, and Deyi Xiong. 2024b. Evaluating chinese large language models on discipline knowledge acquisition via memorization and robustness assessment. In *Proceedings of the 1st Workshop on Data Contamination (CONDA)*, pages 1–12.

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. 2023. Autodan: Generating stealthy jailbreak prompts on aligned large language models. *arXiv preprint arXiv:2310.04451*.

Inbal Magar and Roy Schwartz. 2022. Data contamination: From memorization to exploitation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 157–165.

Pratyush Maini, Hengrui Jia, Nicolas Papernot, and Adam Dziedzic. 2024. Llm dataset inference: Did you train on my dataset? *Advances in Neural Information Processing Systems*, 37:124069–124092.

Behzad Mehrbakhsh, Dario Garigliotti, Fernando Martínez-Plumed, and Jose Hernandez-Orallo. 2024. Confounders in instance variation for the analysis of data contamination. In *Proceedings of the 1st Workshop on Data Contamination (CONDA)*, pages 13–21.

Shiwen Ni, Xiangtao Kong, Chengming Li, Xiping Hu, Ruifeng Xu, Jia Zhu, and Min Yang. 2025. Training on the benchmark is not all you need. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24948–24956.

Yonatan Oren, Nicole Meister, Niladri S Chatterji, Faisal Ladhak, and Tatsunori Hashimoto. 2023. Proving test set contamination in black-box language models. In *The Twelfth International Conference on Learning Representations*.

Medha Palavalli, Amanda Bertsch, and Matthew R Gormley. 2024. A taxonomy for data contamination in large language models. In *Proceedings of the 1st Workshop on Data Contamination (CONDA)*, pages 22–40.

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. 2024. Finetuning aligned language models compromises safety, even when users do not intend to! In *International Conference on Learning Representations*.

Kun Qian, Shunji Wan, Claudia Tang, Youzhi Wang, Xuanming Zhang, Maximillian Chen, and Zhou Yu. 2024. Varbench: Robust language model benchmarking through dynamic variable perturbation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 16131–16161.

Weijia Shi, Anirudh Ajith, Mengzhou Xia, Yangsibo Huang, Daogao Liu, Terra Blevins, Danqi Chen, and Luke Zettlemoyer. 2024. Detecting pretraining data from large language models. In *12th International Conference on Learning Representations, ICLR 2024*.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, and 1 others. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *TRANSACTIONS ON MACHINE LEARNING RESEARCH*.

Liangtai Sun, Yang Han, Zihan Zhao, Da Ma, Zhennan Shen, Baocai Chen, Lu Chen, and Kai Yu. 2024. Scieval: A multi-level large language model evaluation benchmark for scientific research. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19053–19061.

Shangqing Tu, Kejian Zhu, Yushi Bai, Zijun Yao, Lei Hou, and Juanzi Li. 2024. Dice: Detecting in-distribution contamination in llm's fine-tuning phase for math reasoning. *arXiv preprint arXiv:2406.04197*.

Siyuan Wang, Zhuohan Long, Zhihao Fan, Xuan-Jing Huang, and Zhongyu Wei. 2025. Benchmark self-evolving: A multi-agent framework for dynamic llm evaluation. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 3310–3328.

Zeming Wei, Yifei Wang, Ang Li, Yichuan Mo, and Yisen Wang. 2023. Jailbreak and guard aligned language models with only few in-context demonstrations. *arXiv preprint arXiv:2310.06387*.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Benjamin Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, Shubh-Agrawal, Sandeep Singh Sandha, Siddartha Venkat Naidu, Chinmay Hegde, Yann LeCun, Tom Goldstein, Willie Neiswanger, and Micah Goldblum. 2025. Livebench: A challenging, contamination-free LLM benchmark. In *The Thirteenth International Conference on Learning Representations*.

Ruijie Xu, Zengzhi Wang, Run-Ze Fan, and Pengfei Liu. 2024. Benchmarking benchmark leakage in large language models. *arXiv preprint arXiv:2404.18824*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Shuo Yang, Wei-Lin Chiang, Lianmin Zheng, Joseph E Gonzalez, and Ion Stoica. 2023. Rethinking benchmark and contamination for language models with rephrased samples. *arXiv preprint arXiv:2311.04850*.

Feng Yao, Yufan Zhuang, Zihao Sun, Sunan Xu, Animesh Kumar, and Jingbo Shang. 2024. Data contamination can cross language barriers. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17864–17875.

Wentao Ye, Jiaqi Hu, Liyao Li, Haobo Wang, Gang Chen, and Junbo Zhao. 2024. Data contamination calibration for black-box llms. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 10845–10861.

Jiahao YING, Yixin CAO, Yushi BAI, Qianru SUN, Bo WANG, Wei TANG, Zhaojun DING, Yizhe YANG, Xuanjing HUANG, and Shuicheng YAN. 2024. Automating dataset updates towards reliable and timely evaluation of large language models. Conference on Neural Information Processing Systems, NeurIPS.

Zhuohao Yu, Chang Gao, Wenjin Yao, Yidong Wang, Wei Ye, Jindong Wang, Xing Xie, Yue Zhang, and

Shikun Zhang. 2024. Kieval: A knowledge-grounded interactive evaluation framework for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5967–5985.

Jingyang Zhang, Jingwei Sun, Eric Yeats, Yang Ouyang, Martin Kuo, Jianyi Zhang, Hao Frank Yang, and Hai Li. 2025. Min-k%++: Improved baseline for pre-training data detection from large language models. In *The Thirteenth International Conference on Learning Representations*.

Weichao Zhang, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Yixing Fan, and Xueqi Cheng. 2024a. Pretraining data detection for large language models: A divergence-based calibration method. *arXiv preprint arXiv:2409.14781*.

Zhehao Zhang, Jiaao Chen, and Diyi Yang. 2024b. Darg: Dynamic evaluation of large language models via adaptive reasoning graph. *Advances in Neural Information Processing Systems*, 37:135904–135942.

Qihao Zhao, Yangyu Huang, Tengchao Lv, Lei Cui, Qinzheng Sun, Shaoguang Mao, Xin Zhang, Ying Xin, Qiufeng Yin, Scarlett Li, and 1 others. 2024. Mmlu-cf: A contamination-free multi-task language understanding benchmark. *arXiv preprint arXiv:2412.15194*.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2024. Agieval: A human-centric benchmark for evaluating foundation models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2299–2314.

Kaijie Zhu, Jiaao Chen, Jindong Wang, Neil Zhenqiang Gong, Diyi Yang, and Xing Xie. 2024a. Dyval: Dynamic evaluation of large language models for reasoning tasks. In *ICLR*.

Kaijie Zhu, Jindong Wang, Qinlin Zhao, Ruochen Xu, and Xing Xie. 2024b. Dynamic evaluation of large language models by meta probing agents. In *Proceedings of the 41st International Conference on Machine Learning*, pages 62599–62617.

Qin Zhu, Qinyuan Cheng, Runyu Peng, Xiaonan Li, Ru Peng, Tengxiao Liu, Xipeng Qiu, and Xuan-Jing Huang. 2024c. Inference-time decontamination: Reusing leaked benchmarks for large language model evaluation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 9113–9129.

Wenhong Zhu, Hongkun Hao, Zhiwei He, Yun-Ze Song, Jiao Yueyang, Yumeng Zhang, Hanxu Hu, Yiran Wei, Rui Wang, and Hongyuan Lu. 2024d. Clean–eval: Clean evaluation on contaminated large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 835–847.

A Supplementary Explanations of AdEval

A.1 Mechanisms for Mitigating Contamination Risks in Seed Data and Pipeline of AdEval

To address the issue that 'seed data (a subset of the original evaluation set) may be contaminated', this paper avoids contamination transfer from three aspects through the full-link design of 'knowledge point and main idea extraction → Online Search → question design based on cognitive hierarchy → Complexity Control' in the AdEval process: 1) 'Deconcretization' processing of seed data: The extracted 'knowledge points' (such as operator rules of regular expressions) and 'main ideas' (such as 'finding equivalent regular expressions') are all abstract core concepts, rather than specific expressions or cases in the original questions. Even if the original seed questions are contaminated, the abstraction process has stripped away 'concrete information that can be memorized by the model', retaining only the cognitive goals required for evaluation; 2) The 'auxiliary positioning' role of online search: Online search is only used to supplement background information and examples of knowledge points (such as application scenarios of regular expressions), and is not the only source of question design. Question generation must be anchored to 'abstract knowledge points + core main ideas' at the same time, and must follow Bloom's cognitive hierarchy (such as multi-dimensional design rules from 'remembering' to 'creating'). Even if the search content is contaminated, it will be filtered by the triple constraints of 'knowledge points - main ideas - cognitive dimensions' and cannot directly determine the form of the questions;

3) 'Difference guarantee' of dynamic generation: In the question design phase, through rules such as 'avoiding repetition with original questions', 'distinguishable option logic', and 'unique answers', combined with subsequent 'iterative complexity adjustment' and 'multi-model vote check', the expression differences and logical independence between the final data and the original seed questions as well as the contaminated search content are further ensured, blocking contamination transfer at the result level.

In summary, AdEval does not rely on the 'direct reuse' of seed data or search content, but through the process of 'abstract extraction → multi-source constraint generation → multi-round verification',

it disassembles the contamination risks into various links, and finally realizes the anti-contamination ability of dynamically evaluated data."

A.2 Fairness of AdEval

AdEval ensures the fairness of evaluations across different LLMs through "unified dynamic dataset generation". The core of its fairness lies in "providing the same dynamically generated dataset for all models to be evaluated", so as to ensure that different types and capabilities of LLMs (such as open-source small-parameter model Llama3-8B and closed-source large-parameter model GPT-4) are under the same evaluation benchmark, and avoid evaluation bias caused by data differences. The generation process of the dynamic dataset is independent of the models to be evaluated. AdEval only takes the "knowledge points and main ideas of the static benchmark (knowledge point and main idea extraction)" as the anchor (e.g., extracting core cognitive goals such as "judgment of regular expression equivalence" from MMLU), combines the background information from Online Search and the rules of Bloom's cognitive hierarchy, and generates a unique set of dynamic dataset. The generation process does not rely on any model capability information (such as model parameter scale and historical performance), and will not "customize and adjust data difficulty or content" for specific models; at the same time, through iterative reconstruction (e.g., adjusting the difficulty of questions based on the accuracy feedback of Qwen2-7B-Chat), the complexity of the dynamic dataset is aligned with the original static benchmark (with a deviation $<$ 5%). This complexity alignment mechanism can not only prevent the complete failure of small models' performance due to overly difficult data, but also avoid the saturation of large models' performance due to overly simple data, ensuring that the dataset has effective discrimination for models of different capabilities and further ensuring the fairness of evaluations across different LLMs.

A.3 Time Cost Analysis of AdEval's Dynamic Dataset Generation

This section analyzes and discusses the time cost of dynamic dataset generation. Taking a single original static data as the unit, this section quantifies the total time consumption of AdEval throughout the entire process from knowledge point extraction to quality check. The logic of AdEval generating dynamic data from a single original data is "1 original data $\rightarrow$ 3 knowledge points $\rightarrow$ 6 Bloom's cognitive hierarchy questions corresponding to each knowledge point", and finally 18 dynamic data are generated (1×3×6=18).

The time consumption of knowledge point extraction is approximately 4 seconds; the time consumption of main idea extraction is approximately 4 seconds; for online search, 1 online search is conducted for each of the 3 knowledge points, with a time consumption of approximately 3×15=45 seconds; for Bloom's cognitive hierarchy question design, questions are divided into 3 groups according to knowledge points, and 6 questions are generated in each group, with a time consumption of approximately 3×6×5=90 seconds; approximately 5 rounds of complexity alignment are required, and about half of the questions need to be sampled for reconstruction in each round, taking approximately 5×(18×0.5)×5=225 seconds; for quality check and reconstruction, 18 questions are inspected, and approximately 6% of the questions may have quality issues and require reconstruction, taking approximately (18×4)+(18×0.06×5)=77.4 seconds. Therefore, the total time for generating 18 valid dynamic data from a single original data is approximately: 4+4+45+90+225+77.4=445.4 seconds, so the generation time of one dynamic data is approximately 445.4/4=24.7 seconds.

B More Detailed Related Work

B.1 Static Evaluation Benchmarks for Large Language Models

The evaluation of Large Language Models (LLMs) relies heavily on various benchmark datasets, among which static evaluation benchmarks have laid the foundation for assessing model capabilities. These benchmarks cover multiple disciplines, task types, and evaluation dimensions, providing a standardized basis for comparing different models. For example, MMLU (Massive Multitask Language Understanding) (Hendrycks et al., 2020) is a representative benchmark for multi-disciplinary evaluation, used to assess models' knowledge mastery and reasoning abilities in different fields. HELM (Holistic Evaluation of Language Models) (Liang et al., 2022) expands the scope of evaluation from basic performance to dimensions such as fairness, efficiency, and robustness, providing a more comprehensive evaluation framework. BIG-Bench (Srivastava et al., 2022) breaks the boundaries of model capability testing through 204 challenging tasks,

exploring the performance limits of LLMs in professional and complex scenarios. For Chinese-language models, AGIEval (Zhong et al., 2024) and C-Eval (Huang et al., 2023) are two core benchmarks. AGIEval focuses on standardized examination content, while C-Eval emphasizes the evaluation of advanced Chinese knowledge; the two together meet the needs of assessing the localized capabilities of models. In the field of scientific reasoning, ARC-Challenge (Clark et al., 2018) specifically assesses models' scientific knowledge reserves and logical reasoning abilities. BELLE (Ji et al., 2023), on the other hand, focuses on the core capabilities of natural language processing and evaluates both the understanding and generation performance of models. However, static benchmarks have fundamental limitations—they are vulnerable to data contamination. Since datasets remain fixed once constructed, as the scale of models expands, the problem of overlap between these benchmarks and the massive pre-training data of LLMs (often sourced from the same online resources) becomes increasingly prominent. This overlap may lead to overestimation of model capabilities, produce misleading conclusions, and seriously undermine the fairness and effectiveness of model comparisons.

B.2 Data Contamination

Data contamination has become a critical bottleneck in the evaluation of LLMs (Deng et al., 2024; Cheng et al., 2025). Different from the characteristic that training and test data can be easily separated in traditional NLP and machine learning tasks, the scale of LLM pre-training data is huge (usually reaching tens of terabytes), making it difficult to achieve complete isolation from evaluation benchmarks. When test samples or their variants exist in the pre-training corpus, the model may rely on memory rather than real reasoning to generate correct responses, resulting in distorted capability evaluation. Existing studies have confirmed the multi-dimensional impacts of data contamination: Magar and Schwartz (2022) and Jiang et al. (2024) found that contamination in the pre-training phase significantly affects model performance, and factors such as data volume and model scale will adjust the degree of contamination; Palavalli et al. (2024) and Li and Flanigan (2024) further confirmed that contamination will distort the evaluation results of downstream tasks as well as zero-shot and few-shot learning capabilities; more alarmingly, the "Evasion through Augmented Learning (EAL)" technology proposed by Dekoninck et al. (2024b) can bypass most existing detection methods to achieve contamination, which highlights the urgency of the problem.

B.3 Data Contamination Detection Methods

Data contamination detection methods in the past three years include detecting contamination by changing the data order (e.g., PAC(Ye et al., 2024), Oren et al. (2023), Ni et al. (2025)); using perplexity-based detection (e.g., Xu et al. (2024), Li (2023)); analyzing LLMs' internal state distributions or identifying low-probability outlier tokens (e.g., DICE (Tu et al., 2024), Min-K%++(Zhang et al., 2025), Shi et al. (2024)); and identifying contamination by detecting and correcting output distributions (e.g., CDD and TED (Dong et al., 2024)). Other methods, such as DetCon and ComiEval (Dong et al., 2024), improve benchmark evaluation, while ConStat (Dekoninck et al., 2024a) identifies contamination by comparing benchmark performances. Golchin and Surdeanu (2024) combine guided prompts with classifiers for detection, Yao et al. (2024) propose cross-lingual contamination detection, and Maini et al. (2024) analyze membership characteristics using statistical inference. Although these methods have their own innovations, they generally have problems such as relying on the model's internal information (e.g., activation states) or having complex analysis processes, which limits their application in practical evaluation scenarios.

To quantify and compare data contamination detection technologies, a variety of dedicated evaluation benchmarks have been proposed. Their common feature is that they label text-label pairs based on release dates and establish a temporal evaluation framework. Both WikiMIA and BookMIA were proposed by Shi et al. (2024), which use Wikipedia and book data respectively to evaluate the ability of Membership Inference Attack (MIA) methods to detect pre-training data leakage; MIMIR a Python package launched by Duan et al. (2024), which provides more challenging memory evaluation tasks than WikiMIA; the benchmark PatentMIA proposed by Zhang et al. (2024a), which is specifically designed for Chinese-language pre-training data contamination detection; the StackMIAsub dataset proposed by Ye et al. (2024), which supports white-box and black-box model testing and provides a general platform for the evaluation of detection methods.

B.4 Mitigation Strategies for Data Contamination

B.4.1 Contamination-free Benchmark Construction Strategy

To address the limitations of static benchmarks and detection methods, the following strategies are included: contamination-free benchmark construction and dynamic evaluation.

Contamination-free benchmark construction strategies include but are not limited to: data updating-based methods, data rewriting-based methods, and prevention-based methods. **Data updating-based methods** use the latest data to construct benchmarks, minimizing overlap with historical pre-training data while ensuring stable task difficulty. LatestEval (Li et al., 2024c) implements an automated pipeline to generate contamination-free test sets from recent materials; LiveBench (White et al., 2025) extends this concept to multi-task scenarios, integrating math, coding, and reasoning tasks with dynamic updates and automated scoring. In specialized fields, LiveCodeBench (Jain et al., 2024) focuses on code generation and self-repair capabilities, maintaining regular update cycles, while EvoCodeBench (Li et al., 2024a) dynamically aligns with the latest code repositories to achieve fair coding evaluation; NPHardEval4V (Fan et al., 2024) further adapts this method to the assessment of reasoning capabilities of multimodal LLMs. **Data rewriting-based methods** leverage the paraphrasing and verification capabilities of LLMs to generate samples with equivalent semantics but different vocabularies, thereby eliminating contamination. Clean-Eval (Zhu et al., 2024d) realizes benchmark purification through paraphrasing and back-translation; MMLU-CF (Zhao et al., 2024) enhances reliability by combining diverse question collection, difficulty sampling, LLM-aided integrity checks, and strategies such as question rephrasing and option shuffling; CLEVA (Li et al., 2023) adopts non-repetitive sampling and multi-strategy rewriting for robust evaluation, while (YING et al., 2024) update benchmarks through style-preserving mimicry and cognitive level expansion based on Bloom's taxonomy. **Prevention-based methods** ensure data integrity through technical and procedural controls, with core measures including public-key encryption of test data, strict access permission management, and prohibition of derivative data generation; the Inference-Time Decontamination (ITD) (Zhu et al., 2024c) technique proposes an innovative post-hoc strategy to identify and rewrite potential memorized responses during model inference; the bilingual benchmark C2LEVA (Li et al., 2024b) builds a systematic contamination defense mechanism through proactive measures such as test data rotation and enhanced encryption.

B.4.2 Dynamic Evaluation Strategy

Dynamic evaluation fundamentally solves the contamination problem by generating adaptive and non-repetitive test samples, which can be divided into rule-based dynamic evaluation and agent-based dynamic evaluation. **Rule-based dynamic evaluation** generates controllable and diverse samples, DYVAL (Zhu et al., 2024a) adopts a Directed Acyclic Graph (DAG) architecture to construct multi-step reasoning tasks with precisely adjustable complexity, and its upgraded version DyVal 2 (Zhu et al., 2024b) further enhances sample controllability; S3EVAL (Lei et al., 2024) focuses on SQL evaluation and assesses long-text reasoning capabilities through randomized table-query pairs. DARG (Zhang et al., 2024b) and functionalization techniques expand this direction: DARG uses adaptive reasoning graphs to achieve complexity control, while functionalization transforms static question-answer pairs into parameterized code to generate infinite variants; Qian et al. (2024) introduce variable perturbation technology to realize the controllable variation generation of dynamic datasets; SciEval (Sun et al., 2024) combines static and dynamic data to achieve multi-level evaluation of scientific capabilities, and Liu et al. (2024b) develop comprehensive dynamic tests for Chinese knowledge assessment. **Agent-based dynamic evaluation** utilizes LLM-based agents to realize the autonomous evolution of benchmarks, Multi-Principle Assessment (MPA) (Zhu et al., 2024b) automatically transforms existing questions into new ones through agents, and the multi-agent framework proposed by Wang et al. (2025) realizes the self-evolution of benchmarks by dynamically mutating the context and structure of questions. In addition to generative methods, other optimization strategies reduce contamination by improving existing data: Yang et al. (2023) filter similar samples to reduce the risk of overfitting; DCQ (Golchin and Surdeanu, 2025) generates multiple-choice questions to detect and mitigate leakage; KIEval (Yu et al., 2024) uses multi-turn dialogues to evaluate comprehension and knowledge application capabilities while reduc-

ing memory effects. These methods collectively provide diverse strategies for dynamic evaluation, significantly improving the fairness and reliability of assessment.

C Prompts

C.1 Knowledge Point Extraction - Prompt

You are a professional expert in summarizing knowledge points from questions. Please summarize the relevant knowledge points from the given multiple-choice question.

Requirements:

1. Provide professional knowledge points related to the question.

2. Focus on summarizing broad concepts or categories of knowledge related to the question, rather than specific detailed explanations.

3. Strictly follow the format below and do not output any other content:

["knowledge1", "knowledge2" , "knowledge3", . . .]

{{few-shot}}

Please summarize the following multiple-choice question:

{{choiceQ}}

The term {{few-shot}} refers to few-shot learning, with specific details provided in Appendix D.1. {{choiceQ}} refers to the questions in the dataset. Both of these are embedded in the prompt and provided as input to the large language model.

C.2 Main Idea Extraction - Prompt

You are a professional expert in summarizing the main idea of questions. Please summarize the main idea based on the input question content. Requirements:

1. Provide the main idea related to the question, covering the background of the question and the core of the answer.

2. Ensure the main idea is concise and clear, avoiding redundant expressions.

3. Output only the main idea without any additional content.

{{few-shot}}

Please summarize the main idea of the following question:

{{choiceQ}}

C.3 Online Search - Prompt

You are an expert in explaining knowledge points. Based on the provided question, knowledge point, and main idea, use online resources to search for and provide a detailed explanation closely related to the core of the knowledge point. Please strictly follow these requirements for your output:

1. Focus on the Knowledge Point: The explanation must revolve tightly around the input knowledge point. While the question and main idea can provide background or supplementary context, the knowledge point must remain the core focus.

2. Concise Paragraph: Present the explanation in a single, clear, and coherent paragraph. Avoid excessive length or overly brief statements, and do not include additional structuring or embellishments. 3. Direct and Comprehensive: Ensure the explanation is accurate, logically sound, and centered on the topic, avoiding irrelevant information or ambiguous statements.

{{few-shot}}

Please search online and provide a single-paragraph detailed explanation of the knowledge point, referencing the question and main idea where relevant.

Actual Input Question:

{{choiceQ}}

Actual Input Knowledge Point:

{{kn}}

Actual Input Main Idea:

{{purport}}

{{kn}} and {{purport}} represent the knowledge points and main ideas extracted in C.1 and C.2.

C.4 Question Design - Prompt

You are a professional question designer specializing in creating challenging multiple-choice questions. Based on the provided reference question, knowledge points, main ideas, and text content, please generate one multiple-choice question and its answer. Ensure the question's difficulty is not lower than the reference question, and preferably slightly higher. The question should test students' in-depth understanding, detailed analysis, and comprehensive judgment skills. The multiple-choice question must meet the following requirements:

1. High Relevance: The question, correct option, and answer must be directly based on the provided text and closely related to the main idea.

2. Consistent or Higher Difficulty: The complexity of the question must match or exceed that of the reference question. Avoid simple factual recitations; instead, require students to achieve a deep understanding and analyze details to deduce the correct answer.

3. Reasonable and Distinguishable Options: The options must be logically clear, avoiding overly obvious correct answers. Include at least one strong distractor to test detailed reading and reasoning skills.

4. Avoid external dependency: The question and its options must be entirely based on the provided text content and main idea. If quoting content is necessary, directly embed the relevant text in the question rather than using references like "according to the passage." Avoid creating questions that depend on context not provided, such as: "Which of the following best describes the purpose of penetration testing as described in the text?"

5. Unique Answer: Ensure only one option is correct, while other options remain plausible but subtly incorrect or mismatched with the main idea and text content.

6. Deep Understanding: The question can involve inference, comparison, method analysis, application of key concepts, etc., to increase the difficulty level.

7. Innovative Question Stem: The question format should differ significantly from the reference question, showcasing creativity while maintaining a rigorous and professional style.

8. Different Focus: The question's focus should differ from the reference question. For example, if the reference question emphasizes concept definition, the generated question could focus on method application, key detail comparison, pros and cons analysis, or judgment in practical scenarios.

9. Focus on knowledge points: The question must center around the provided knowledge points, with its core content closely aligned with the main idea.

10. Strict Output Format: The output must strictly follow the JSON structure below, without any additional content:

```
[
  {
    "Question": "Question content",
    "A": "Option content",
    "B": "Option content",
    "C": "Option content",
    "D": "Option content",
    "Answer": "A/B/C/D"
  }
]
```

{{few-shot}}

Below are the provided reference question, knowledge points, main idea, and text content. Please generate a multiple-choice question and answer that meets the above requirements. Remember, do not make the question similar to the reference question, but ensure the difficulty level is consistent with or even higher than the reference question.

Reference question:

{{choiceQ}}

Knowledge Point:

{{kn}}

Main Idea:

{{purport}}

Text Content:

{{KNexplain}}

{{KNexplain}} refers to the detailed explanations of the knowledge points extracted in C.3.

C.5 Bloom's Cognitive Hierarchy Design - Prompt

You are a professional question designer specializing in creating challenging multiple-choice questions. Based on the provided reference question, knowledge points, main ideas, and text content, please generate six multiple-choice questions and answers corresponding to Bloom's six levels of cognitive learning. Ensure the question's difficulty is not lower than the reference question, and preferably slightly higher. The question should test students' in-depth understanding, detailed analysis, and comprehensive judgment skills.

Let me first explain what Bloom's six cognitive levels are:

Introduction to Bloom's Six Cognitive Levels:

Level 1: Remembering

This involves recognizing and recalling concepts and knowledge, storing them in the brain, and retrieving them when needed. Examples include memorizing vocabulary, poetry, or definitions. This level involves identifying concrete or abstract knowledge and can pertain to factual, conceptual, procedural, or metacognitive knowledge. While mechanical, it is a foundational step for learning and solving more complex problems.

Level 2: Understanding

Understanding means grasping the essence of information or knowledge, though not deeply or thoroughly—just a preliminary understanding. When learners connect "new" knowledge to existing knowledge, they achieve understanding. Examples include the classic Feynman technique, where teaching others reinforces personal understanding. Understanding occurs when new information integrates with existing cognitive frameworks.

Understanding includes translating, interpreting, and inferring:

Interpreting involves explaining or summarizing information in one's own words. Translating involves expressing learned content in different forms, such as using diagrams to explain a concept.

Inferring involves predicting trends or developments based on learned knowledge.

Level 3: Applying

This level focuses on applying learned concepts, principles, or rules in new situations or solving real-world problems. It involves correctly using abstract ideas in appropriate contexts without explicit problem-solving instructions. Applications rely on remembering and understanding as foundational, including applying concepts, principles, methods, and theories, such as using the Pythagorean theorem to solve geometry problems.

Level 4: Analyzing

Analyzing involves breaking down complex knowledge into its components and understanding the relationships between them. It requires deconstructing material to identify essential elements and their relationships, making the material's organization and structure clearer. Tasks include identifying, analyzing, and recognizing the interrelationships and structure among concepts.

For instance, discovering how Factor A changes alongside Factor B by analyzing correlations, causal relationships, mediating variables, and moderators.

Level 5: Evaluating

This level involves making judgments about the value of something based on rational and persuasive reasoning rather than intuition or observation. It requires synthesizing internal and external evidence and making objective, well-supported assessments. For example, legal debates where both sides cite laws to support their arguments exemplify evaluation.

Level 6: Creating

This involves reorganizing learned knowledge or generating new information to form a new whole, such as proposing new ideas, solutions, or designs. It requires analyzing and synthesizing components to form a new structure. Creativity emphasizes innovation and problem-solving, breaking conventional thought patterns and achieving the ultimate learning goal.

Creating is the highest level as it challenges learners to develop new knowledge structures, highlighting innovation and originality.

Multiple-Choice Question Requirements:

1. High Relevance: The question, correct option, and answer must be directly based on the provided text and closely related to the main idea.

2. Consistent or Higher Difficulty: The complexity of the question must match or exceed that of the reference question. Avoid simple factual recitations; instead, require students to achieve a deep understanding and analyze details to deduce the correct answer.

3. Reasonable and Distinguishable Options: The options must be logically clear, avoiding overly obvious correct answers. Include at least one strong distractor to test detailed reading and reasoning skills.

4. Avoid external dependency: The question and its options must be entirely based on the provided text content and main idea. If quoting content is necessary, directly embed the relevant text in the question rather than using references like "according to the passage." Avoid creating questions that depend on context not provided, such as: "Which of the following best describes the purpose of penetration testing as described in the text?"

5. Unique Answer: Ensure only one option is correct, while other options remain plausible but subtly incorrect or mismatched with the main idea and text content.

6. Focus on knowledge points: The question must center around the provided knowledge points, with its core content closely aligned with the main idea.

7. Coverage of Levels: Each question must correspond to a single cognitive level, covering all six levels of Bloom's taxonomy, with one question per level. A total of six questions is required.

8. Strict Output Format: The output must strictly follow the JSON structure below, without any additional content:

```
[
```

```
  {
    "Layer": "Cognitive Level",
    "Question": "Question content",
    "A": "Option content",
    "B": "Option content",
    "C": "Option content",
    "D": "Option content",
    "Answer": "A/B/C/D"
  }
  ...
]
```

{{few-shot}}

Below are the provided reference question, knowledge points, main idea, and text content. Please generate Bloom's six cognitive-level multiple-choice questions that meets the above requirements. Remember, do not make the question similar to the reference question, but ensure the difficulty level is consistent with or even higher than the reference question.

Reference question:

{{choiceQ}}

Knowledge Point:

{{kn}}

Main Idea:

{{purport}}

Text Content:

{{KNexplain}}

C.6 Complexity Control - Prompt

You are a professional question reconstruction expert, specializing in enhancing question complexity and designing challenging multiple-choice questions.

Please redesign a multiple-choice question based on the provided original question, current question, knowledge points, main idea, and text content. The goal is to improve students' in-depth understanding, detailed analysis, and comprehensive judgment skills.

Multiple-Choice Question Design Requirements:

1. High Relevance: The stem, correct option, and distractors must be strictly based on the provided text content. Ensure the question is closely related to the main idea, and the answer can be explicitly supported by the text.

2. Increased Difficulty: The question should be more complex than the current one. Avoid simple factual repetition; instead, require students to deeply understand, analyze details, and make connections to arrive at the correct answer.

3. Reasonable and Distinguishable Options: The options must be logically clear, avoiding overly obvious correct answers. Include at least one strong distractor to test detailed reading and reasoning skills.

4. Avoid external dependency: The question and its options must be entirely based on the provided text content and main idea. If quoting content is necessary, directly embed the relevant text in the question rather than using references like "according to the passage." Avoid creating questions that depend on context not provided, such as: "Which of the following best describes the purpose of penetration testing as described in the text?"

5. Unique Answer: Ensure only one option is correct, while other options remain plausible but subtly incorrect or mismatched with the main idea and text content.

6. Deep Understanding: The question can involve inference, comparison, method analysis, application of key concepts, etc., to increase the difficulty level.

7. Innovative Question Stem: The question format should differ significantly from the original question, showcasing creativity while maintaining a rigorous and professional style.

8. Different Focus: The question's focus should differ from the original question. For example, if the original question emphasizes concept definition, the generated question could focus on method application, key detail comparison, pros and cons analysis, or judgment in practical scenarios.

9. Focus on knowledge points: The question must center around the provided knowledge points, with its core content closely aligned with the main idea. 10. Strict Output Format: The output must strictly follow the JSON structure below, without any additional content:

```
[
  {
    "Question": "Question content",
    "A": "Option content",
    "B": "Option content",
    "C": "Option content",
    "D": "Option content",
    "Answer": "A/B/C/D"
  }
]
```

{{few-shot}}

The following includes the original question, current question, knowledge points, main idea, and text content. Please generate a multiple-choice question that meets the above requirements and provide an answer. Be sure not to make it similar to the original question, and ensure that the difficulty of the newly generated question is higher than that of the current question.

Original question:

{{choiceQ}}

current question:

{{choiceQCurrent}}

Knowledge Point:

{{kn}}

Main Idea:

{{purport}}

Text Content:

{{KNexplain}}

{{choiceQCurrent}} refers to the questions dynamically generated in C.4 and C.5.

C.7 Quality Control - Prompt

C.7.1 Question Answer Checking Prompt

You are a professional multiple-choice question proofreading expert, specializing in verifying the accuracy of questions and their answers.

I will provide a multiple-choice question and its answer. Please strictly follow the rules below to make your judgment and respond only in numeric form:

1. If the answer is incorrect, respond with "0".
2. If the answer is correct, respond with "1".
3. Your response must be either 0/1, with no additional content.

Example Input:

Which of the following best describes the balance the Supreme Court has struck between the establishment clause and the free-exercise clause? A Freedom of speech is protected except in certain situations, such as yelling "fire" in a crowded theater. B Once a church has been recognized by the federal government, its tax-exempt status can never be revoked. C Once Congress has created an administrative agency, that agency can be dissolved only by a constitutional amendment. D State-sponsored prayer during school hours is prohibited, but voluntary prayer by student groups before school is allowed.

Answer:D

Example Output:

0

Below is a multiple-choice question and its answer. Please judge according to the rules and respond only with 0/1:

{{choiceQ}}

D Few-Shot

D.1 Knowledge Point Extraction - Few-Shot

Example Input1:

"Question: Which of the following regular expressions is equivalent to (describes the same set of strings as) (a* + b)*(c + d)?

A a*(c + d)+ b(c + d).

B a*(c + d)* + b(c + d)*.

C a*(c + d)+ b*(c + d).

D (a + b)*c +(a + b)*d

Answer:D"

Example Output1:

["Regular expressions and their operators", "Concatenation in regular expressions", "Union operator (+) in regular expressions", "Kleene star (*) in regular expressions", "Equivalence of regular expressions", "Pattern matching with regular expressions"]

Example Input2:

"Question: A certain pipelined RISC machine has 8 general-purpose registers R0, R1, . . . , R7 and supports the following operations. ADD Rs1, Rs2, Rd Add Rs1 to Rs2 and put the sum in Rd

MUL Rs1, Rs2, Rd Multiply Rs1 by Rs2 and put the product in Rd

An operation normally takes one cycle; however, an operation takes two cycles if it produces a result required by the immediately following operation in an operation sequence. Consider the expression AB + ABC + BC, where variables A, B, C are located in registers R0, R1, R2. If the contents of these three registers must not be modified, what is the minimum number of clock cycles required for an operation sequence that computes the value of AB + ABC + BC?"

A 5 B 6 C 7 D 8

Answer:B"

Example Output2:

["RISC architecture and instruction pipelining", "Data hazards and their impact on instruction execution", "Register allocation and usage in assembly language", "Instruction scheduling to minimize clock cycles", "Arithmetic operations in assembly language"]

D.2 Main Idea Extraction - Few-Shot

Example Input1:

"Question: Which of the following regular expressions is equivalent to (describes the same set of strings as) (a* + b)*(c + d)?

A a*(c + d)+ b(c + d).

B a*(c + d)* + b(c + d)*.

C a*(c + d)+ b*(c + d).

D (a + b)*c +(a + b)*d

Answer:D"

Example Output1:

Finding the regular expression equivalent to (a* + b)*(c + d).

Example Input2:

"Question: A certain pipelined RISC machine has 8 general-purpose registers R0, R1, . . . , R7 and supports the following operations. ADD Rs1, Rs2, Rd Add Rs1 to Rs2 and put the sum in Rd MUL Rs1, Rs2, Rd Multiply Rs1 by Rs2 and put the product in Rd An operation normally takes one cycle; however, an operation takes two cycles if it produces a result required by the immediately following operation in an operation sequence. Consider the expression AB + ABC + BC, where variables A, B, C are located in registers R0, R1, R2. If the contents of these three registers must not be modified, what is the minimum number of clock cycles required for an operation sequence that computes the value of AB + ABC + BC?"

A 5 B 6 C 7 D 8

Answer:B"

Example Output2:

Determining the minimum number of clock cycles required to compute the expression $AB + ABC + BC$ on a pipelined RISC machine, considering operation latencies, data dependencies, and constraints on register modification.

D.3 Online Search - Few-Shot

Example Input1:

Question:

Which of the following regular expressions is equivalent to (describes the same set of strings as) (a* + b)*(c + d)?

A a*(c + d)+ b(c + d).

B a*(c + d)* + b(c + d)*.

C a*(c + d)+ b*(c + d).

D (a + b)*c +(a + b)*d

Answer:D

Knowledge Point:

Regular expressions and their operators

Main Idea: Finding the regular expression equivalent to (a* + b)*(c + d).

Example Output1:

Regular expressions are sequences of characters that define search patterns, primarily used for pattern matching within strings. The fundamental operators in regular expressions include concatenation (placing expressions sequentially), alternation (denoted by '|' or '+', representing a choice between expressions), and the Kleene star (denoted by '*', indicating zero or more repetitions of the preceding element). Understanding these operators is crucial for constructing expressions that accurately represent specific string patterns. For instance, the expression '(a|b)*' denotes any combination of the characters 'a' and 'b', including the empty string, while 'a*' matches any number of consecutive 'a' characters, including none. Mastery of these operators enables the creation of complex patterns for efficient text processing and matching tasks.

Example Input2:

Question: A certain pipelined RISC machine has 8 general-purpose registers R0, R1, . . . , R7 and supports the following operations. ADD Rs1, Rs2, Rd Add Rs1 to Rs2 and put the sum in Rd MUL Rs1, Rs2, Rd Multiply Rs1 by Rs2 and put the product in Rd An operation normally takes one cycle; however, an operation takes two cycles if it produces a result required by the immediately following operation in an operation sequence. Consider the expression AB + ABC + BC, where variables A, B, C are located in registers R0, R1, R2. If the contents of these three registers must not be modified, what is the minimum number of clock cycles required for an operation sequence that computes the value of AB + ABC + BC?"

A 5 B 6 C 7 D 8

Answer:B

Knowledge Point:

Register allocation and usage in assembly language

Main Idea:

Determining the minimum number of clock cycles required to compute the expression $AB + ABC + BC$ on a pipelined RISC machine, considering operation latencies, data dependencies, and constraints on register modification.

Example Output2:

In assembly language programming, efficient register allocation and usage are crucial for optimizing performance, especially in pipelined RISC architectures. Registers are limited, high-speed

storage locations within the CPU that hold data for immediate processing, reducing the need to access slower main memory. Effective register allocation involves assigning these registers to frequently used variables and intermediate results, minimizing memory access and enhancing execution speed. This process can be complex due to the limited number of registers and the need to manage data dependencies and instruction latencies. Techniques such as graph coloring are often employed to allocate registers efficiently by modeling interference between variables and ensuring that no two simultaneously live variables share the same register. Additionally, understanding the specific roles and constraints of registers in a given architecture is essential, as certain instructions may require operands to reside in specific registers. By carefully managing register allocation and usage, assembly language programmers can significantly improve the efficiency and performance of their code.

D.4 Question Design - Few-Shot

Example Input1:

Reference question:

Which of the following regular expressions is equivalent to (describes the same set of strings as) (a* + b)*(c + d)?

A a*(c + d)+ b(c + d).

B a*(c + d)* + b(c + d)*.

C a*(c + d)+ b*(c + d).

D (a + b)*c +(a + b)*d

Answer:D

Knowledge Point:

Regular expressions and their operators

Main Idea:

Finding the regular expression equivalent to (a* + b)*(c + d).

Text Content:

Regular expressions are sequences of characters that define search patterns, primarily used for pattern matching within strings. The fundamental operators in regular expressions include concatenation (placing expressions sequentially), alternation (denoted by '|' or '+', representing a choice between expressions), and the Kleene star (denoted by '*', indicating zero or more repetitions of the preceding element). Understanding these operators is crucial for constructing expressions that accurately represent specific string patterns. For instance, the expression '(a|b)*' denotes any combination of the characters 'a' and 'b', including the empty string, while 'a*' matches any number of consecutive 'a' characters, including none. Mastery of these operators enables the creation of complex patterns for efficient text processing and matching tasks.

Example Output1:

[{

"Question": "Which of the following regular expressions correctly represents all strings that start with zero or more repetitions of either 'x' or 'y', followed by exactly two repetitions of either 'm' or 'n', and end with a single occurrence of either 'p' or 'q'?",

"A": "$(x|y) * ((m|n)(m|n))(p|q)$",

"B": "$((x|y) * (m|n))^2(p|q)$",

"C": "$(x * +y*)((m + n)(m + n))(p + q)$",

"D": "$(x|y)^2((m|n)(m|n))(p|q)$",

"Answer": "A"

}]

Example Input2:

Reference question:

A certain pipelined RISC machine has 8 general-purpose registers R0, R1, . . . , R7 and supports the following operations. ADD Rs1, Rs2, Rd Add Rs1 to Rs2 and put the sum in Rd MUL Rs1, Rs2, Rd Multiply Rs1 by Rs2 and put the product in Rd An operation normally takes one cycle; however, an operation takes two cycles if it produces a result required by the immediately following operation in an operation sequence. Consider the expression AB + ABC + BC, where variables A, B, C are located in registers R0, R1, R2. If the contents of these three registers must not be modified, what is the minimum number of clock cycles required for an operation sequence that computes the value of AB + ABC + BC?"

A 5 B 6 C 7 D 8

Answer:B

Knowledge Point:

Register allocation and usage in assembly language

Main Idea:

Determining the minimum number of clock cycles required to compute the expression $AB + ABC + BC$ on a pipelined RISC machine, considering operation latencies, data dependencies, and constraints on register modification.

Text Content:

In assembly language programming, efficient register allocation and usage are crucial for optimizing performance, especially in pipelined RISC architectures. Registers are limited, high-speed storage locations within the CPU that hold data

for immediate processing, reducing the need to access slower main memory. Effective register allocation involves assigning these registers to frequently used variables and intermediate results, minimizing memory access and enhancing execution speed. This process can be complex due to the limited number of registers and the need to manage data dependencies and instruction latencies. Techniques such as graph coloring are often employed to allocate registers efficiently by modeling interference between variables and ensuring that no two simultaneously live variables share the same register. Additionally, understanding the specific roles and constraints of registers in a given architecture is essential, as certain instructions may require operands to reside in specific registers. By carefully managing register allocation and usage, assembly language programmers can significantly improve the efficiency and performance of their code.

Example Output2:

[{

"Question": "Given the same pipelined RISC machine and the expression $AB + ABC + BC$, where A, B, and C are in registers R0, R1, and R2 respectively, and assuming no register modification is allowed, which sequence of operations would result in the minimum number of clock cycles, and what is that minimum number?",

"A": "Perform MUL R0, R1, R3; MUL R1, R2, R4; ADD R3, R4, R5; ADD R5, R2, R6; 7 cycles",

"B": "Perform MUL R0, R1, R3; ADD R3, R1, R4; MUL R1, R2, R5; ADD R4, R5, R6; 7 cycles",

"C": "Perform MUL R0, R1, R3; MUL R3, R2, R4; ADD R3, R2, R5; ADD R4, R5, R6; 6 cycles",

"D": "Perform MUL R0, R1, R3; MUL R1, R2, R4; ADD R3, R4, R5; ADD R5, R1, R6; 6 cycles",

"Answer": "C"

}]

Finally, remember not to output in the following format (do not begin with

```

```

[{ "Question": "Which of the following regular expressions correctly represents all strings that can be formed by zero or more repetitions of the strings made up of 'a' or 'b', followed by exactly one occurrence of either 'c' or 'd'?",

"A": "$(a|b) * (c|d)$",

"B": "$(a * b * +b * a*)(c|d)$",

"C": "$(a + b) * (c|d)+$",

"D": "$((a + b) * c) + ((a + b) * d)$",

"Answer": "A"

}]

```

### D.5 Bloom's Cognitive Hierarchy Question Design - Few-Shot

Example Input1:

Reference question:

When trying to construct a regular expression to match strings that start with an optional single occurrence of either 'e' or 'f', followed by zero or more repetitions of a group containing either 'g' or 'h' and then either 'i' or 'j', and end with exactly three repetitions of either 'k' or 'l', which of the following options would be the correct regular expression?

A $(e|f)?((g|h) * (i|j))(k|l)^3$

B $((e|f) + (g|h)) * (i|j)(k|l)^3$

C $(e|f)(g|h) * (i|j)^3(k|l)^3$

D $(e|f) * ((g|h) + (i|j))(k|l)^3$

Answer: A

Knowledge Point:

Regular expressions and their operators

Main Idea:

Finding the regular expression equivalent to (a* + b)*(c + d).

Text Content:

Regular expressions are sequences of characters that define search patterns, primarily used for pattern matching within strings. The fundamental operators in regular expressions include concatenation (placing expressions sequentially), alternation (denoted by 'l' or '+', representing a choice between expressions), and the Kleene star (denoted by '*', indicating zero or more repetitions of the preceding element). Understanding these operators is crucial for constructing expressions that accurately represent specific string patterns. For instance, the expression '(alb)*' denotes any combination of the characters 'a' and 'b', including the empty string, while 'a*' matches any number of consecutive 'a' characters, including none. Mastery of these operators enables the creation of complex patterns for efficient text processing and matching tasks.

Example Output1:

[ {

"Layer": "Remembering", "Question": "Which symbol in regular expressions denotes zero or more repetitions of the preceding element?",

"A": "?", "B": "*", "C": "+", "D": ".",
```

"Answer": "B"
}, { "Layer": "Understanding",
"Question": "In the context of regular expressions, what does the expression ‘(a|b)*‘ represent?",
"A": "Any string that contains only ’a’ repeated any number of times",
"B": "Any string that contains only ’b’ repeated any number of times",
"C": "Any combination of ’a’ and ’b’, including the empty string",
"D": "Any string that starts with ’a’ or ’b’ and ends with ’a’ or ’b’",
"Answer": "C"
},
{ "Layer": "Applying",
"Question": "Given the regular expression ‘(a* + b)*(c + d)‘, which of the following regular expressions correctly applies the same pattern?",
"A": "$(a|b) * (c|d)$",
"B": "$(a + b) * (c + d)$",
"C": "$(a * b) * (c + d)$",
"D": "$(a + b) * (c|d)$",
"Answer": "D"
},
{
"Layer": "Analyzing",
"Question": "How does the regular expression ‘(a|b)*‘ differ from ‘a*‘ in terms of the possible strings it can match?",
"A": "They match exactly the same set of strings",
"B": "‘(a|b)*‘ can match strings with ’b’, while ‘a*‘ cannot",
"C": "‘a*‘ can match strings with ’b’, while ‘(a|b)*‘ cannot",
"D": "They both match strings with any number of ’a’s, including ’b’s",
"Answer": "B"
},
{
"Layer": "Evaluating",
"Question": "Considering the regular expression ‘(a* + b)*(c + d)‘, evaluate which of the following statements best justifies the use of the alternation operator ‘|‘ over the concatenation operator ‘+‘ in this context?",
"A": "The ‘|‘ operator allows for a choice between ’a’ and ’b’, which is necessary for matching a pattern that could start with either ’a’ or ’b’",
"B": "The ‘+‘ operator is used for one or more repetitions, which is not suitable for optional elements",
"C": "The ‘|‘ operator is more efficient for matching a single character",
"D": "The ‘|‘ operator is used to match either ’a’ or ’b’, but it is not necessary for the pattern to be valid",
"Answer": "A"
},
{
"Layer": "Creating",
"Question": "If you were to create a new regular expression based on the pattern ‘(a* + b)*(c + d)‘ that matches a string starting with zero or more ’a’s, followed by one or more ’b’s, and ending with either ’c’ or ’d’, what would that regular expression be?",
"A": "(a*b+)(c|d)",
"B": "(a+b*)(c|d)",
"C": "(a* + b+)(c|d)",
"D": "(a+b*)(c+d)",
"Answer": "C"
}]
Finally, remember not to output in the following format (do not begin with
```
“‘json
[ {
"Layer": "Remembering",
"Question": "Which symbol in regular expressions denotes zero or more repetitions of the preceding element?",
"A": "?", "B": "*", "C": "+", "D": ".",
"Answer": "B"
}, ]
```

D.6 Complexity Control - Few-Shot

Example Input1:
Original question:
Question: Which of the following regular expressions is equivalent to (describes the same set of strings as) (a* + b)*(c + d)?
A a*(c + d)+ b(c + d).
B a*(c + d)* + b(c + d)*.
C a*(c + d)+ b*(c + d).
D (a + b)*c +(a + b)*d
Answer:D
current question:

Which of the following regular expressions correctly represents all strings that start with zero or more repetitions of either 'x' or 'y', followed by exactly two repetitions of either 'm' or 'n', and end with a single occurrence of either 'p' or 'q'?

A $(x|y) * ((m|n)(m|n))(p|q)$

B $((x|y) * (m|n))^2(p|q)$

C $(x * +y*)((m + n)(m + n))(p + q)$

D $(x|y)^2((m|n)(m|n))(p|q)$

Answer:A

Knowledge Point:

Regular expressions and their operators

Main Idea:

Finding the regular expression equivalent to (a* + b)*(c + d).

Text Content:

Regular expressions are sequences of characters that define search patterns, primarily used for pattern matching within strings. The fundamental operators in regular expressions include concatenation (placing expressions sequentially), alternation (denoted by '|' or '+', representing a choice between expressions), and the Kleene star (denoted by '*', indicating zero or more repetitions of the preceding element). Understanding these operators is crucial for constructing expressions that accurately represent specific string patterns. For instance, the expression '(a|b)*' denotes any combination of the characters 'a' and 'b', including the empty string, while 'a*' matches any number of consecutive 'a' characters, including none. Mastery of these operators enables the creation of complex patterns for efficient text processing and matching tasks.

Example Output1:

[{ "Question": "When trying to construct a regular expression to match strings that start with an optional single occurrence of either 'e' or 'f', followed by zero or more repetitions of a group containing either 'g' or 'h' and then either 'i' or 'j', and end with exactly three repetitions of either 'k' or 'l', which of the following options would be the correct regular expression? ",

"A": "$(e|f)?((g|h) * (i|j))(k|l)^3$",

"B": "$((e|f) + (g|h)) * (i|j)(k|l)^3$",

"C": "$(e|f)(g|h) * (i|j)^3(k|l)^3$",

"D": "$(e|f) * ((g|h) + (i|j))(k|l)^3$",

"Answer": "A" }]

Example Input2:

Original question:

Question: A certain pipelined RISC machine has 8 general-purpose registers R0, R1, . . . , R7 and supports the following operations. ADD Rs1, Rs2, Rd Add Rs1 to Rs2 and put the sum in Rd MUL Rs1, Rs2, Rd Multiply Rs1 by Rs2 and put the product in Rd An operation normally takes one cycle; however, an operation takes two cycles if it produces a result required by the immediately following operation in an operation sequence. Consider the expression AB + ABC + BC, where variables A, B, C are located in registers R0, R1, R2. If the contents of these three registers must not be modified, what is the minimum number of clock cycles required for an operation sequence that computes the value of AB + ABC + BC?"

A 5 B 6 C 7 D 8

Answer:B

current question:

Given the same pipelined RISC machine and the expression $AB + ABC + BC$, where A, B, and C are in registers R0, R1, and R2 respectively, and assuming no register modification is allowed, which sequence of operations would result in the minimum number of clock cycles, and what is that minimum number?

A Perform MUL R0, R1, R3; MUL R1, R2, R4; ADD R3, R4, R5; ADD R5, R2, R6; 7 cycles

B Perform MUL R0, R1, R3; ADD R3, R1, R4; MUL R1, R2, R5; ADD R4, R5, R6; 7 cycles

C Perform MUL R0, R1, R3; MUL R3, R2, R4; ADD R3, R2, R5; ADD R4, R5, R6; 6 cycles

D Perform MUL R0, R1, R3; MUL R1, R2, R4; ADD R3, R4, R5; ADD R5, R1, R6; 6 cycles

Answer: C

Knowledge Point:

Register allocation and usage in assembly language

Main Idea:

Determining the minimum number of clock cycles required to compute the expression $AB + ABC + BC$ on a pipelined RISC machine, considering operation latencies, data dependencies, and constraints on register modification.

Text Content:

In assembly language programming, efficient register allocation and usage are crucial for optimizing performance, especially in pipelined RISC architectures. Registers are limited, high-speed storage locations within the CPU that hold data for immediate processing, reducing the need to access slower main memory. Effective register allocation involves assigning these registers to frequently used variables and intermediate results, minimizing memory access and enhancing execution speed. This process can be complex due to

the limited number of registers and the need to manage data dependencies and instruction latencies. Techniques such as graph coloring are often employed to allocate registers efficiently by modeling interference between variables and ensuring that no two simultaneously live variables share the same register. Additionally, understanding the specific roles and constraints of registers in a given architecture is essential, as certain instructions may require operands to reside in specific registers. By carefully managing register allocation and usage, assembly language programmers can significantly improve the efficiency and performance of their code.

Example Output2:

[{

"Question": "When computing the expression (AB + ABC + BC) on the described pipelined RISC machine, if we consider not only the operation latencies and register modification constraints but also the impact of register allocation strategies like graph coloring on reducing data dependencies, which of the following operation sequences would be the most efficient in terms of minimizing the number of clock cycles while also ensuring the best utilization of registers considering potential interference between variables according to the principles of register allocation in assembly language, and what is that minimum number of cycles?",

"A": "Perform MUL R0, R1, R3; then use graph coloring to reallocate R1 for another operation to avoid interference; MUL R3, R2, R4; ADD R3, R2, R5; ADD R4, R5, R6; 5 cycles",

"B": "First apply graph coloring to allocate registers optimally for the initial MUL operations; perform MUL R0, R1, R3; MUL R1, R2, R4; then handle ADD operations considering data dependencies based on the coloring result; ADD R3, R4, R5; ADD R5, R2, R6; 6 cycles",

"C": "Without using graph coloring, just follow a traditional register allocation based on operand order; perform MUL R0, R1, R3; MUL R3, R2, R4; ADD R3, R2, R5; ADD R4, R5, R6; 7 cycles",

"D": "Use graph coloring but in a suboptimal way that leads to some unnecessary register swaps; perform MUL R0, R1, R3; then swap R1 with another register due to the coloring plan; MUL R1, R2, R4; ADD R3, R4, R5; ADD R5, R2, R6; 8 cycles",

"Answer": "B"

}]

Finally, remember not to output in the following format (do not begin with

```

```

[{ "Question": "When trying to construct a regular expression to match strings that start with an optional single occurrence of either 'e' or 'f', followed by zero or more repetitions of a group containing either 'g' or 'h' and then either 'i' or 'j', and end with exactly three repetitions of either 'k' or 'l', which of the following options would be the correct regular expression? ",

"A": "$(e|f)?((g|h)*(i|j))(k|l)^3$",

"B": "$((e|f)+(g|h))*(i|j)(k|l)^3$",

"C": "$(e|f)(g|h)*(i|j)^3(k|l)^3$",

"D": "$(e|f)*((g|h)+(i|j))(k|l)^3$",

"Answer": "A" }]

```

## E Question Design Example Based on Bloom's cognitive hierarchy

**Layer: Remembering**

Question: Which symbol in regular expressions denotes zero or more repetitions of the preceding element?

A: ? B: * C: + D: .

Answer: B

**Layer: Understanding**

Question: In the context of regular expressions, what does the expression '(a|b)*' represent?

A: Any string that contains only 'a' repeated any number of times

B: Any string that contains only 'b' repeated any number of times

C: Any combination of 'a' and 'b', including the empty string

D: Any string that starts with 'a' or 'b' and ends with 'a' or 'b'

Answer: C

**Layer: Applying**

Question: Given the regular expression '(a* + b)*(c + d)', which of the following regular expressions correctly applies the same pattern?

A: (a|b)*(c|d) B: (a+b)*(c+d)

C: (a*b)*(c+d) D: (a+b)*(c|d)

Answer: D

**Layer: Analyzing**

Question: How does the regular expression '(a|b)*' differ from 'a*' in terms of the possible strings it can match?

A: They match exactly the same set of strings
```

B: ‘(a|b)*‘ can match strings with ’b’, while ‘a*‘ cannot

C: ‘a*‘ can match strings with ’b’, while ‘(a|b)*‘ cannot

D: They both match strings with any number of ’a’s including ’b’s

Answer: B

Layer: Evaluating

Question: Considering the regular expression ‘(a* + b)*(c + d)‘, evaluate which of the following statements best justifies the use of the alternation operator ‘|‘ over the concatenation operator ‘+‘ in this context?

A: The ‘|‘ operator allows for a choice between ’a’ and ’b’, which is necessary for matching a pattern that could start with either ’a’ or ’b’

B: The ‘+‘ operator is used for one or more repetitions, which is not suitable for optional elements

C: The ‘|‘ operator is more efficient for matching a single character

D: The ‘|‘ operator is used to match either ’a’ or ’b’, but it is not necessary for the pattern to be valid

Answer: A

Layer: Creating

Question: If you were to create a new regular expression based on the pattern ‘(a* + b)*(c + d)‘ that matches a string starting with zero or more ’a’s, followed by one or more ’b’s, and ending with either ’c’ or ’d’, what would that regular expression be?,

A: (a*b+)(c|d) B: (a+b*)(c|d)

C: (a* + b+)(c|d) D: (a+b*)(c+d)

Answer: C

F Comparison of examples of dynamic dataset generation methods

Static

An astronomer observes that a planet rotates faster after a meteorite impact. Which is the most likely effect of this increase in rotation?

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

AdEval

A planet experiences a significant increase in its rotation speed due to a series of meteorite impacts. If the planet’s initial angular momentum was L_1 and its final angular momentum after the impacts is L_2, and the initial length of a planetary day was T_1 and the final length is T_2, which of the following relationships correctly describes the change in angular momentum and the change in the length of the planetary day, assuming no other significant external torques act on the planet?

A $L_2 > L_1$ and $T_2 < T_1$

B $L_2 < L_1$ and $T_2 > T_1$

C $L_2 > L_1$ and $T_2 > T_1$

D $L_2 < L_1$ and $T_2 < T_1$

Answer: A

Auto-Dataset

An astronaut notices that a celestial body spins more rapidly after a large asteroid collision. What is the most probable consequence of this enhanced rotation?

A The mass of the celestial body will increase.

B The seasons on the celestial body will last longer.

C The length of a day on the celestial body will be reduced.

D The magnetic field of the celestial body will weaken.

Answer: C

LatestEaval According to the principle of conservation of angular momentum, when a meteorite impacts a planet and imparts additional angular momentum, what is the most likely immediate effect?

A The density of the planet increases.

B The length of the planetary year decreases.

C The length of the planetary day becomes shorter.

D The gravity of the planet increases.

Answer: C

G LLaMA-Factory Fine-tunes LLMs

The loss values during the fine-tuning of Qwen2-7B-Chat and Llama3-8B-Instruct using LLaMA-Factory are shown in Figure 7 and Figure 8. It can be seen that due to the small amount of experimental data, the loss values approach 0 after 35 iterations, and at the same time, the loss values decrease at the fastest rate during the first 10 iterations.

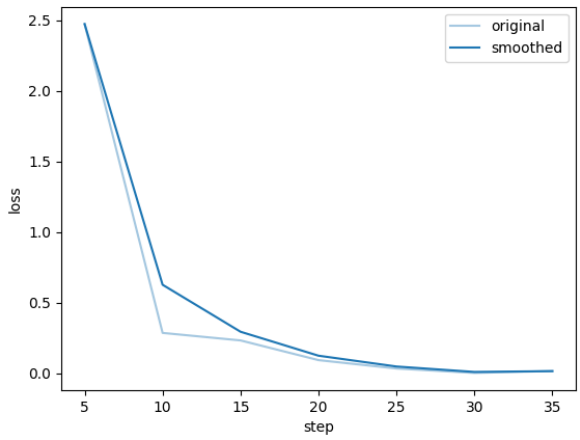

Figure 7: The loss of Qwen2-7B-Chat trained in LLaMA-Factory

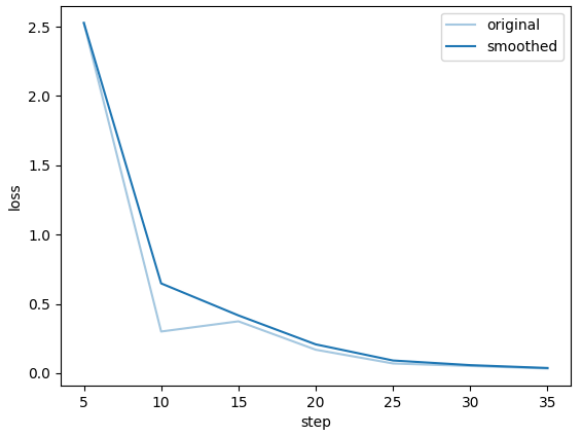

Figure 8: The loss of Llama3-8B-Instruct trained in LLaMA-Factory