

Complex Instruction with LLM Paper Survey

Large-scale language models (LLM) have demonstrated outstanding zero-shot and few-shot capabilities in a large number of NLP tasks, which not only broadens the scope of application in the NLP field but also poses a huge challenge to the ability to evaluate LLM. Therefore, this prompts us to think:

- Where is the boundary of LLM's ability to complete complex tasks?
- Which tasks are complex for LLM and why?

Starting from the perspective of exploring the complex and new tasks that have been completed using LLM to complete various existing NLP, this survey comprehensively sorts out the complex tasks that LLM is difficult to handle based on the paper that comprehensively evaluates LLM's ability, thereby constructing the complex task characteristics relative to LLM, and providing clear guidance for us to evaluate LLM.

1. Complete complex tasks with LLM

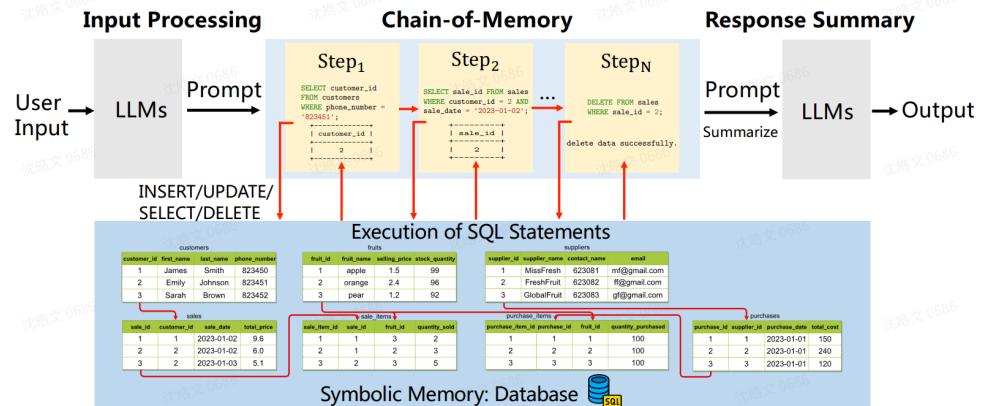
In this section, we investigate the relevant complex tasks currently being completed using LLM, in order to gain a preliminary understanding of the complex and difficult tasks currently facing the LLM community.

1.1 Complex logical reasoning tasks

Recently, researchers have begun to verify and analyze the ability of LLM to solve complex logical reasoning tasks (definition of logical reasoning ability: the ability to decompose tasks and solve them one by one). Some representative papers containing complex logical reasoning tasks are listed below.

CHATDB: AUGMENTING LLMS WITH DATABASES AS THEIR SYMBOLIC MEMORY	<p>This paper mainly combines databases and LLM, uses databases as the symbolic memory of LLM, and solves the problem of context length limitation by operating symbolic memory</p> <ul style="list-style-type: none">• This article first analyzes the limitations of current LLM in long texts, especially long texts caused by multiple rounds of dialogue, which will make LLM forget or lose control of dialogue history.
--	--

- In response to this problem, the paper proposes the CoM (Chain of Memory) method to operate symbolic memory: by converting user instructions (complex tasks) into a series of intermediate operations on database memory, it solves complex problems related to variable storage and memory.

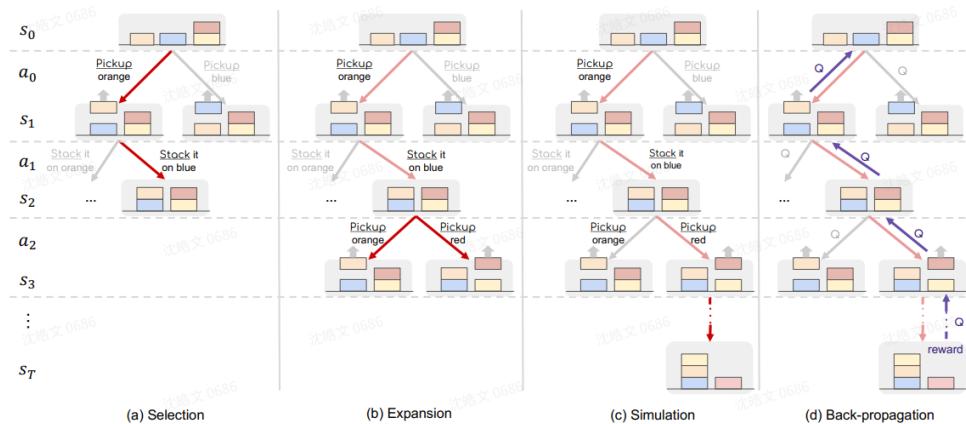


This method does not require any memory information (intermediate reasoning information) to be saved in the prompt, but stored in the database, so the memory is more accurate and direct. The effect on the synthesized Fruit Shop Dataset is significant due to ChatGPT.

Model	Easy	Hard	All	Accuracy
ChatGPT	10/15	1/35	11/50	22%
ChatDB (ours)	13/15	28/35	41/50	82%

Reasoning with Language Model is Planning with World Model

The paper argues that a complex task often requires multi-step reasoning, and the reasoning process requires maintaining an internal hidden state, namely a world model, which can be used to predict the state of the environment and store intermediate variables to simulate the long-term results of actions. This paper proposes a RAP reasoning framework, which explicitly models the state prediction, reward prediction, and action prediction of complex tasks by combining LLM and MCTS Monte Carlo tree search.



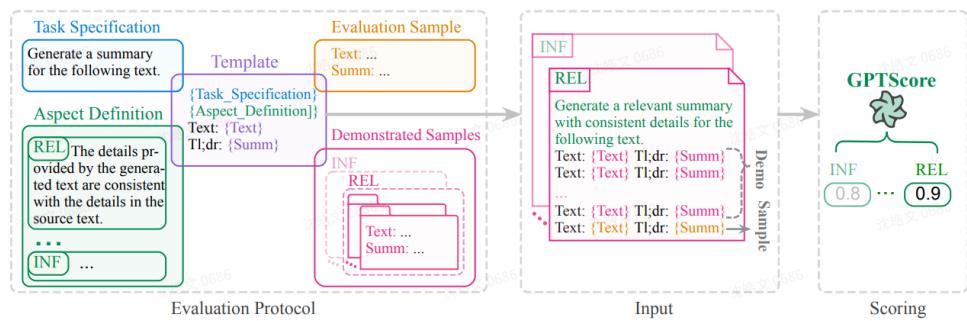
This article conducted sufficient testing on the following three datasets and found that maintaining a world model can effectively improve LLM's ability to solve complex problems.

- Plan generation (Blocksworld benchmark): Spatial inference dataset
- Math reasoning (GSM8K)
- Logical Inference (PrOntoQA): Define a series of rules and a fact to be tested, and verify the correctness of the fact through logical reasoning

1.2 Data Annotation (Evaluation), Analysis, and Correction Tasks

With the continuous development of LLM, the quality of replies and texts generated using LLM is increasingly approaching the quality of real text. For specific tasks, evaluation indicators designed specifically such as BLEU can be used to accurately evaluate accuracy. However, for free format and high entropy tasks such as open text generation, text summarization, and casual conversations, it is often difficult to automatically measure the quality of generated replies. With the continuous development of LLM technology, currently, using LLM to complete analysis, evaluation, and correction tasks related to text generation has become a more mainstream evaluation method. Next, we will introduce some papers that use LLM to complete evaluation tasks.

This article is the first to propose an automatic evaluation method for text generation-related tasks using LLM.



GPTScore: Evaluate as You Desire

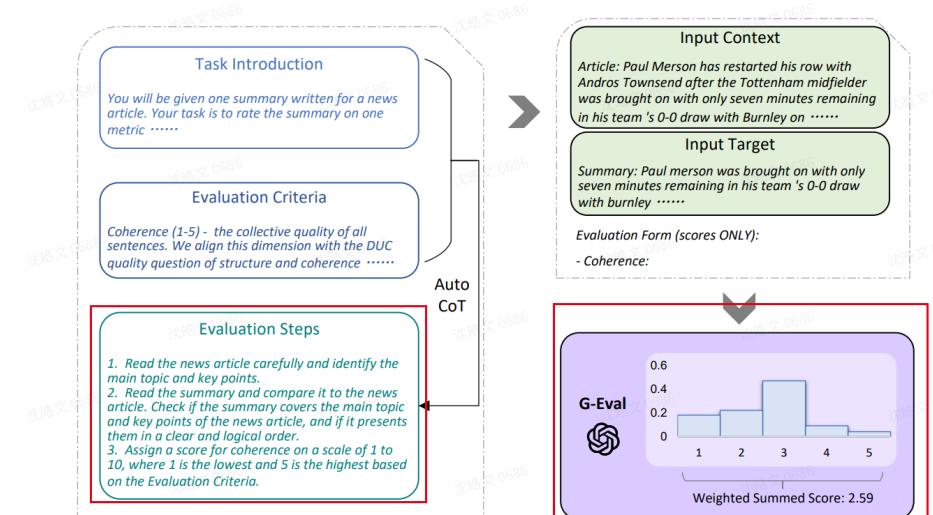
Unlike previous learning-based automatic evaluation indicators, the advantages of using LLM's evaluation indicators are:

1. No training required
2. Quickly migrate to new tasks and scenarios with prompt designing
3. Multi-language support, etc.

The core contribution of this article is to verify that LLM can achieve the same or even better human evaluation consistency than current learning-based automatic evaluation indicators on a large number of text generation tasks, and to verify that introducing few-shot demonstration and detailed evaluation indicator introduction and definition can significantly improve the relevance of evaluation.

G-EVAL: NLG Evaluation using GPT-4 with Better Human Alignment

Unlike existing work (before G-Eval and after GPTScore), G-EVAL verified that using CoT can help LLM understand evaluation tasks and make more accurate evaluation conclusions.



The key trick used is:

- Use CoT to automatically generate precautions in evaluation
- Use form-filling to gradually score each evaluation dimension
- Weighting the scores of 1-5 points in the final output using the probability of the corresponding token

This article does not pursue improvement in effect, but distills an evaluation model from GPT-4/ChatGPT. By using Alpaca to construct data samples containing evaluation_result/evaluation_reason/ground_truth, LLM is trained to complete the corresponding evaluation tasks.

```
"inputs": {  
    "instruction": "Find an example of the given kind of data",  
    "input": "Qualitative data",  
    "response1": "An example of qualitative data is customer feedback.",  
    "response2": "An example of qualitative data is a customer review."  
}  
  
"outputs": {  
    "evaluation_result": "Tie",  
    "evaluation_reason": "Both responses are correct and provide similar examples of qualitative data.",  
    "reference_response": "An example of qualitative data is an interview transcript."  
}
```

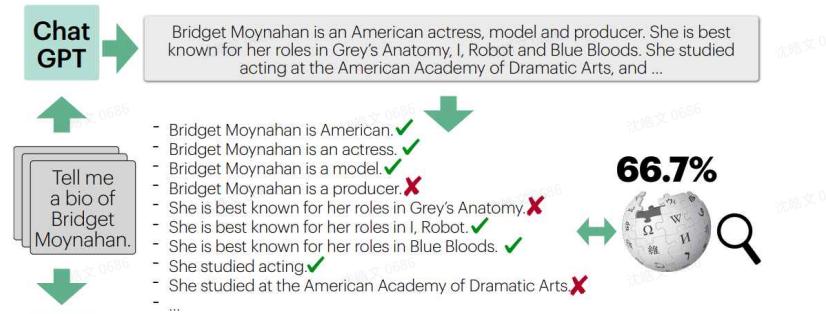
However, the limitation of this method is that the evaluation tasks are too open and lack evaluation data collection for specific domain tasks, making it difficult to perform performance close to ChatGPT/GPT-4 in specific evaluation tasks.

PandaLM: ReProducible and Automated Language Model Assessment

FActScore: Fine-grained Atomic Evaluation of Factual Precision in Long Form Text Generation

The text is mainly aimed at a special evaluation task in open text evaluation, namely factual consistency evaluation. The purpose of this task is to determine how likely the text generated by LLM is to be factually consistent (i.e. without the problem of Hallucination). This article constructs an interpretable factual consistency evaluation scheme and proposes an atomized factual consistency evaluation framework, which is divided into two steps

- Use LLM to break down the statements that generate content, and use LLM to break down multiple evaluated sub-statements from the input text
- Use human evaluation or LLM-based methods to evaluate the factual consistency of statements one by one, and finally evaluate the consistency of the text to be evaluated by calculating the proportion of consistent statements (precision)



Is ChatGPT a Good Teacher Coach? Measuring Zero-Shot Performance For Scoring and Providing Actionable Insights on Classroom Instruction

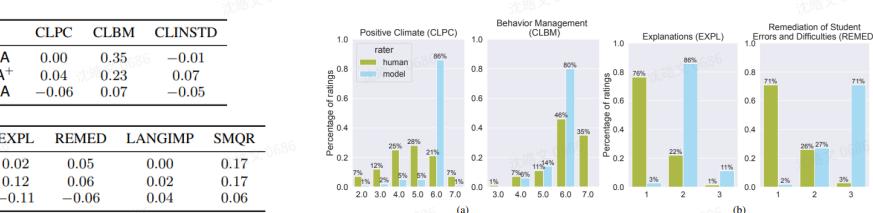
This article comprehensively analyzes the ability of ChatGPT to complete automatic labeling **in the US junior high school teaching field tasks**, and mainly discusses three tasks:

1. Evaluate the teacher's ability in a certain aspect based on the interaction records (transcribed into text) between the teacher and the students in the classroom
2. Analyze the interaction records to identify the points where the teacher did well and the points where they did not do well
3. Analyze interaction records and provide suggestions to teachers

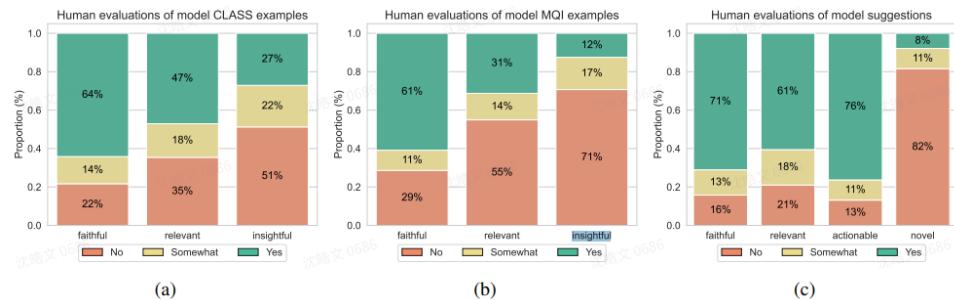
For task 1, the experimental results show that the correlation between ChatGPT and human annotators is poor, and ChatGPT often tends to score high.

	CLPC	CLBM	CLINSTD
DA	0.00	0.35	-0.01
DA ⁺	0.04	0.23	0.07
RA	-0.06	0.07	-0.05

	EXPL	REMED	LANGIMP	SMQR
DA	0.02	0.05	0.00	0.17
DA ⁺	0.12	0.06	0.02	0.17
RA	-0.11	-0.06	0.04	0.06



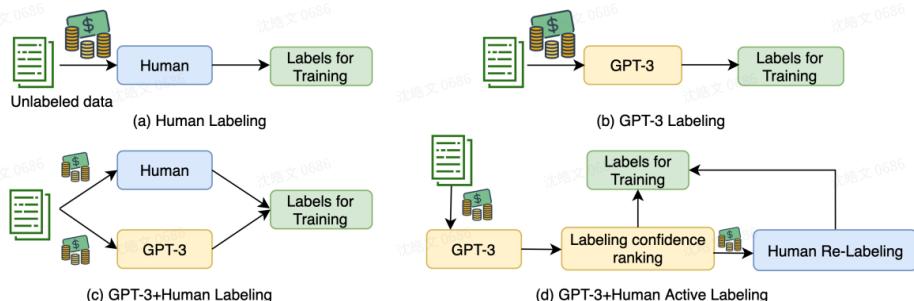
For task 2, the experimental results show that ChatGPT has poor explanatory and evaluative abilities in teaching, especially from the perspective of insightful (Figures a and b), and especially in mathematical abilities (MQI dataset).



For task 3, the experimental results found that the suggestions generated by ChatGPT lack originality (Figure c). In summary, it can be found that ChatGPT lacks sufficient recognition in measuring the creativity of text in such tasks, and there is a serious problem of position bias. This may be due to the lack of specific and niche task data for ChatGPT.

Same as above, this article also uses LLM to complete the task of Data Annotation. A joint annotation scheme of mixed GPT-3 and manual annotation is proposed. The experiment proves the feasibility and effectiveness of LLM in annotating data. The experimental conclusion is as follows:

- Although there is noise in the GPT3 annotation, the problem caused by this noise can be alleviated to some extent by manually re-evaluating the Confidence Level based on logits (active strategies)

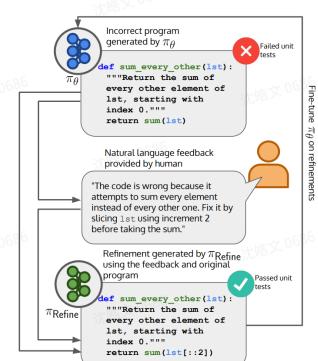


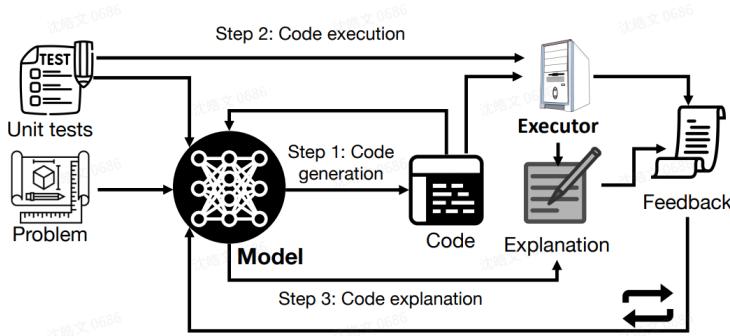
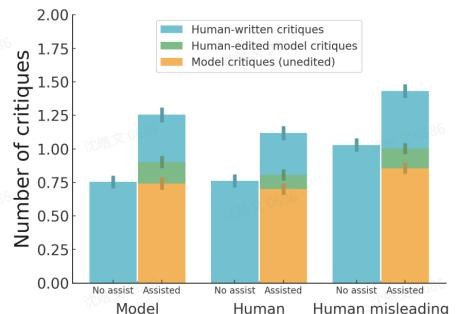
- The model retrained using GPT-3 data can exceed the discriminant results of GPT-3 itself. This phenomenon is theoretically explained in the article (due to the optimization reduction of the upper bound of errors).
- This article tested three generation tasks (summary generation, question generation, title generation) and six classification tasks. Although the tasks are comprehensive, they are still relatively simple.

Want To Reduce Labeling Cost? GPT-3 Can Help

Improving Code Generation by Training with Natural Language Feedback

This article automatically refines the code generated by the model by introducing human feedback (natural language feedback description). This task requires LLM to conduct in-depth analysis and understanding of the completed code to discover potential problems. Based on natural language feedback, LLM also needs to further modify



	<p>the existing code based on the feedback information. The above process of explanation, evaluation, and refinement requires LLM to conduct in-depth analysis of the knowledge in the code field.</p>												
Teaching Large Language Models to Self-Debug	<p>This article is the follow-up work of the above paper. By allowing LLM to automatically generate line-by-line explanatory information in the code, it helps LLM better understand and correct errors by combining the above information as feedback.</p> 												
Self-critiquing models for assisting human evaluators	<p>OpenAI's work has found that language models can assist human annotators in discovering potential errors in model-generated and manually written text summaries by generating natural language descriptions.</p> <p>The experiment found that:</p> <ul style="list-style-type: none"> a. LLM is better at detecting potential errors in text, and as the size of LLM increases, the improvement of helpfulness brought by this ability becomes more obvious b. Although the generated content is difficult enough, LLM still has the ability of self-criticism c. LLM can aggregate the generated feedback information and further improve the quality of text generation  <table border="1"> <thead> <tr> <th>Category</th> <th>No assist</th> <th>Assisted</th> </tr> </thead> <tbody> <tr> <td>Model</td> <td>~0.75</td> <td>~1.25</td> </tr> <tr> <td>Human</td> <td>~0.75</td> <td>~1.15</td> </tr> <tr> <td>Human misleading</td> <td>~1.05</td> <td>~1.45</td> </tr> </tbody> </table>	Category	No assist	Assisted	Model	~0.75	~1.25	Human	~0.75	~1.15	Human misleading	~1.05	~1.45
Category	No assist	Assisted											
Model	~0.75	~1.25											
Human	~0.75	~1.15											
Human misleading	~1.05	~1.45											
WizardLM/WizardCoder	<p>This article uses the idea of instruction evolution to rewrite existing prompts to construct more complex instruction data to complete the Model Training in the SFT stage. By designing prompts, the following data</p>												

rewriting and complexity methods are designed for general text generation:

- Deepening the problem: further thinking and questioning the problem to improve its depth
- Add restrictions: provide more restrictions for the problem, making the solution more complex
- Complex input: Rewrite the problem into a more complex form, such as rewriting a simple mathematical formula into a more complex new formula.
- Concretization: Provide more concrete application scenarios for abstract descriptions to examine LLM's abstract understanding ability
- Increase the reasoning process: By rewriting the current problem into a new problem with more logical reasoning steps, the complexity is increased
- Rewrite as a new question: By rewriting the current question as another question, enhance the diversity of themes, skills, and other aspects.

Under the idea of instruction planning, in the field of code generation, it also introduces

- Code debugging task
- Time and space complexity limit tasks (add restriction information to make the problem more difficult to solve).

1.3 Tasks that LLM still finds difficult to handle

In addition to the complex tasks mentioned above, this section will introduce some cases of using LLM to solve more complex problems. These problems can currently be solved by ChatGPT and GPT-4, and even GPT-4 has some difficulties in processing them.

Sparks of Artificial General Intelligence: Early experiments with GPT-4

This article fully evaluates the ability of GPT-4 model on benchmarks and complex tasks, and lists some interesting settings as follows:

- General capabilities:
 - Generate random images in the style of a painter and do it with javascript code
 - Prove the existence of an infinite number of prime numbers in Shakespeare's writing style
 - Generate a code that outputs the user's diabetes risk based on information such as height and weight
- Code capability

- Use the latest LeetCode code test questions to test the coding ability of GPT-4

	Easy		Median		Hard		Overall	
	$k = 1$	$k = 5$						
pass@k								
GPT-4	68.2	86.4	40.0	60.0	10.7	14.3	38.0	53.0
text-davinci-003	50.0	81.8	16.0	34.0	0.0	3.6	19.0	36.0
Codex (code-davinci-002)	27.3	50.0	12.0	22.0	3.6	3.6	13.0	23.0
Human (LeetCode users)			72.2		37.7		7.0	
								38.2

- Front-end , game development

GPT-4

Prompt:

```
Can you write a 3D game in HTML with Javascript, I want:
-There are three avatars, each is a sphere.
-The player controls its avatar using arrow keys to move.
-The enemy avatar is trying to catch the player.
-The defender avatar is trying to block the enemy.
-There are also random obstacles as cubes spawned randomly at the beginning and moving randomly. The avatars cannot cross those cubes.
-The player moves on a 2D plane surrounded by walls that he cannot cross. The wall should cover the boundary of the entire plane.
-Add physics to the environment using cannon.
-If the enemy catches the player, the game is over.
-Plot the trajectories of all the three avatars.
```

- Deep learning related task development

GPT-4 vs ChatGPT

Prompt:

```
I want to write a customized optimizer in pytorch, the optimizer should do:
1. the parameters for the optimizer is learning rate, momentum ,weight decay, d_dim, k and factor alpha
2. the optimizer goes through each parameter in the network, and then
-- Get the gradient of this parameter, reshape it to 2D by looping through the dimensions, and keeping the dimension whose actual dimension (shape) is equal to d_dim, do an SVD decomposition of the parameter so  $W = U \Sigma V$ .
-- Keep the top k singular vectors of W, so we get a new matrix  $W_k = U \Sigma_{(k)} V$ .
-- Keep the top 2k singular vectors of W, so we get a new matrix  $W_{(2k)} = U \Sigma_{(2k)} V$ .
-- Normalize  $W_k$  according to the momentum of the F-norm of  $W_{(2k)}$ .
-- Truncate each coordinate of  $W_k$  to the top alpha percentile of all the coordinates, in absolute value.
-- Apply momentum on  $W_k$ , and then update the network parameters using this momentum.
-- Apply weight decay.
```

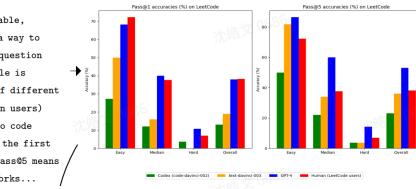
- Convert problematic LaTeX code into the correct code form
- Reverse engineering, testing GPT-4's ability to understand basic computer knowledge such as assembly language and operating systems
- Explain code and predict code execution results, even executable code operations (print all intermediate processes), execute pseudocode
- The technical report of GPT4 also mentions cases of how to use existing library functions and third-party tools to solve complex tasks, such as:
 - Interactive Data Analysis Visualization
 - Deep learning related code construction

GPT-4

Prompt:

[in an interactive manner]

I will show you a LaTeX table, and we will come up with a way to visualize it better in a question and answer format. The table is showing the performance of different language models (and human users) in generating solutions to code challenges. Pass@1 means the first generation works, while Pass@5 means one out of 5 generation works...



```
def step(tensor):
    for group in tensor.grad_fn._grad_fn_iter():
        if group is None:
            continue
        grad = group.grad_fn
        if grad is None:
            continue
        grad_fn = grad.next_functions[0][0]
        if grad_fn is None:
            continue
        grad_fn_type = type(grad_fn)
        if grad_fn_type == torch.autograd.function.FunctionBase:
            grad_fn_type_name = "Function"
        else:
            grad_fn_type_name = "Op"
        if grad_fn_type_name == "Function":
            if grad_fn_type_name == "Function":
                if grad_fn_type_name == "Function":
                    if grad_fn_type_name == "Function":
                        if grad_fn_type_name == "Function":
                            if grad_fn_type_name == "Function":
                                if grad_fn_type_name == "Function":
                                    if grad_fn_type_name == "Function":
                                        if grad_fn_type_name == "Function":
                                            if grad_fn_type_name == "Function":
                                                if grad_fn_type_name == "Function":
                                                    if grad_fn_type_name == "Function":
                                                        if grad_fn_type_name == "Function":
                                                            if grad_fn_type_name == "Function":
                                                                if grad_fn_type_name == "Function":
                                                                    if grad_fn_type_name == "Function":
                                                                        if grad_fn_type_name == "Function":
                                                                            if grad_fn_type_name == "Function":
                                                                                if grad_fn_type_name == "Function":
                                                                                    if grad_fn_type_name == "Function":
                                                                                        if grad_fn_type_name == "Function":
                                                                                            if grad_fn_type_name == "Function":
                                                                                                if grad_fn_type_name == "Function":
                                                                                                 if grad_fn_type_name == "Function":
                                                                                                     if grad_fn_type_name == "Function":
................................................................
```

- Complete a TODO List (including dozens or even hundreds of steps): basic abilities of GPT-Engineer and AutoGPT
- As a discriminant model of factual consistency

Judge	GPT-4	GPT-3	Neither	Both
GPT-4	87.76%	11.01%	1.23%	-
Human	47.61%	6.35%	22.75%	23.29%
Human (constrained)	89.83%	10.07%	-	-

However, there are still some problems that are difficult to handle with models based on autoregressive architecture.

- Text generation tasks that need to be planned in advance: Because GPT-4 cannot backtrack its generation results, some planned content may require multiple iterative corrections. This is difficult for generative models such as GPT-4 to handle.
- Need to store additional memory, information, etc. to assist in reasoning (this problem can be solved in part by CoT)

GPT-4

Q: How many prime numbers are there between 150 and 250?
A: There are 13 prime numbers between 150 and 250.

- Lack of planning for arithmetic and reasoning
- Lack of planning for text generation
 - Local planning: generate prosody and limit the nth word (similar to hidden head, hidden tail, etc.)
 - Global planning, such as

GPT-4

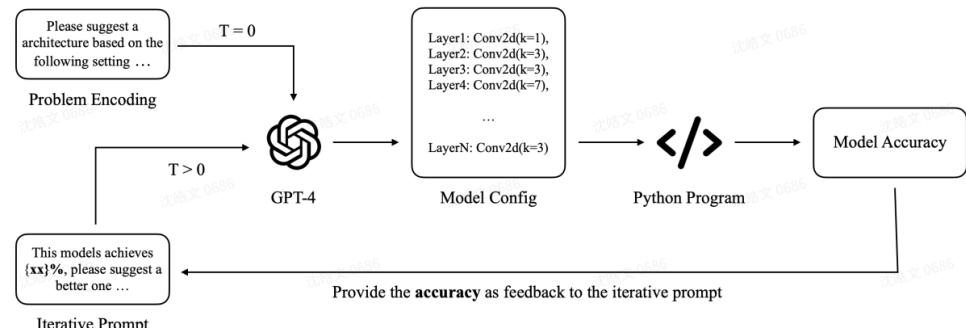
Write a short poem where the last sentence and the first sentence have the same words, but in reverse order. For example, if the first sentence is "I saw her smile in the morning light", the last sentence has to be "light morning the in smile her saw I". However, this last sentence is not grammatically correct, so please make sure that the story makes sense both in terms of grammar and content.
Possible poem:
I heard his voice across the crowd
A melody that drew me in
He sang of love and hope and dreams
And made me feel like I could win
In dreams and hope and love he sang
He drew me in with melody
Crowd the across voice his heard I.

The above problems show that the existing LLM lacks planning, working memory, backtracking and reasoning capabilities, so LLM is very good at handling incremental text generation tasks, such as summarizing, answering questions. But it is difficult to handle some complex tasks that require epiphanies, such as writing a scientific hypothesis, using axioms to solve mathematical proofs, etc

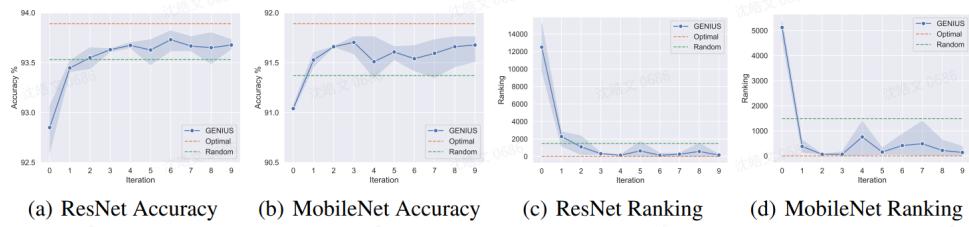
Can GPT-4 Perform Neural Architecture Search?

The author first analyzes that GPT-4 can clearly and effectively explain why ResNet and Transformer have the characteristics and advantages in processing text and image data, which prompts the author to consider whether GPT-4 has enough knowledge and ability to design neural networks architectures. Therefore, this paper studies the use of GPT-4 as a Black box optimizer to perform neural architecture search tasks. The purpose is to design the structure of neural networks to achieve better results on certain tasks, and to optimize the design of the model by

iteratively using GPT-4 to process encoding problems and feedback information.



First, from the preliminary visualization results, GPT-4 is stronger than the random baseline, and a better network structure can be found:



Secondly, from the large-scale evaluation experimental results, the network results of GPT-4 design are basically comparable to the current SOTA NAS technology:

Method	CIFAR10		CIFAR100		ImageNet16-120	
	Validation	Test	Validation	Test	Validation	Test
DARTS [47]	39.77±0.00	54.30±0.00	38.57±0.00	15.61±0.00	18.87±0.00	16.32±0.00
DSNAS [32]	89.66±0.29	93.08±0.13	30.87±16.40	31.01±16.38	40.61±0.09	41.07±0.09
PC-DARTS [77]	89.96±0.15	93.41±0.30	67.12±0.39	67.48±0.89	40.83±0.08	41.31±0.22
SNAS [76]	90.10±1.04	92.77±0.83	69.69±2.39	69.34±1.98	42.84±1.79	43.16±2.64
iDARTS [82]	89.86±0.60	93.58±0.32	70.57±0.24	70.83±0.48	40.38±0.59	40.89±0.68
GDAS [17]	89.89±0.08	93.61±0.09	71.34±0.04	70.70±0.30	41.59±1.33	41.71±0.98
DRNAS [10]	91.55±0.00	94.36±0.00	73.49±0.00	73.51±0.00	46.37±0.00	46.34±0.00
β -DARTS [78]	91.55±0.00	94.36±0.00	73.49±0.00	73.51±0.00	46.37±0.00	46.34±0.00
Λ -DARTS [51]	91.55±0.00	94.36±0.00	73.49±0.00	73.51±0.00	46.37±0.00	46.34±0.00
GENIUS (Ours)	91.07±0.20	93.79±0.09	70.96±0.33	70.91±0.72	45.29±0.81	44.96±1.02

It can be seen that LLM has extensive experience in designing neural networks architectures, but still has the following problems:

- Difficult to reproduce and control
- Benchmark pollution makes it impossible to distinguish whether the existing training data is included in the GPT-4 training corpus

2. The benchmark for evaluating LLM

2.1 Assessing various abilities

The common ability to evaluate LLM is often expanded from the following dimensions. The advantage of the following benchmarks is that they are defined and evaluated using a strictly

agreed-upon evaluation method, so they are simple, interpretable, and easy to reproduce. They are often used to assist in evaluating the general ability of LLM:

different disciplines	<p>We believe that expertise in different fields contributes to a comprehensive examination of the fundamental competencies of LLM</p> <ul style="list-style-type: none">• C-Eval: exam questions for junior high school, high school, university, and different majors (humanities, science and engineering)• MMLU: Multidisciplinary Multiple Choice Questions (Mathematics, History, Science, Law)• AGIEval: college entrance exams, law school exams, math competitions, bar exams• GAOKAO: College entrance examination questions• ARC: Multidisciplinary Examination Dataset•
knowledge utilization	<ul style="list-style-type: none">• BoolQ: Q & A tasks for questions, paragraphs, and answers• CommonsenseQA: Common sense knowledge questions• Natural Questions: Real user Q & A data for Wiki Lingo• TriviaQA: Reading comprehension question answering data•
reasoning ability	<ul style="list-style-type: none">• NLI Related: Three Categories of Natural Language Reasoning Tasks, Implication, Neutral, and Contradiction• HellaSwag: Common Sense Natural Language Reasoning Tasks• GSM8K: Primary Mathematics Question and Answer Data• HumanEval: Code problem-solving dataset, generating code snippets based on problems and code templates
comprehension	<ul style="list-style-type: none">• XSum: Text summarizing data, understanding input documents, generating summaries• MultiRC: Consider multiple pieces of information in context simultaneously to solve problems• LAMBADA: Testing a model's understanding of a document by predicting information about the last word of a document (word prediction)

Although the above benchmarks are very simple and easy to use, in fact, testing other general capabilities of LLM still has limitations, such as the inability to effectively detect LLM's logical reasoning ability, long text modeling ability, combinatorial generalization ability, etc. Therefore,

researchers have also proposed some other benchmarks for more comprehensive analysis and measurement of LLM's general capabilities. Some representative papers are listed below:

<h3>Can ChatGPT Understand Too? A Comparative Study on ChatGPT and Fine-tuned BERT</h3>	<p>This article is a report of ChatGPT testing its NLU capability on GLUE benchmark just after it was released. The experimental conclusions are as follows:</p> <ul style="list-style-type: none"> • ChatGPT is better than fine-tuned BERT in NLI Inference • ChatGPT is similar to fine-tuned BERT in sentiment analysis and QA • ChatGPT struggles with paraphrasing and semantic similarity tasks <ul style="list-style-type: none"> ◦ Negative paraphrase recognition is poor in the Paraphrase task ◦ Neutral similarity samples in NLI 																																																																																																																																																																																																																											
<h3>Are Large Language Models Really Good Logical Reasoners? A Comprehensive Evaluation and Beyond</h3>	<p>This paper tests four kinds of logical derivations and makes formal definitions</p> <ul style="list-style-type: none"> • Deductive reasoning: presup1 + rules -> presup2 • Inductive reasoning: presup1 + presup2 -> rules • Abductive reasoning: presup2 + rule -> presup1 • Hybrid reasoning: involves multiple reasoning abilities (similar to breadth tests). <p>Experiments show that there is still a big gap between the existing LLM's ability to derive the above logic and the current SOTA method:</p> <table border="1" data-bbox="493 1258 1461 1594"> <thead> <tr> <th rowspan="2"></th> <th rowspan="2">Dataset</th> <th rowspan="2">Gen.</th> <th colspan="3">text-davinci-003</th> <th colspan="3">ChatGPT</th> <th colspan="3">BARD</th> <th rowspan="2">SOTA</th> </tr> <tr> <th>0-shot</th> <th>1-shot</th> <th>3-shot</th> <th>0-shot</th> <th>1-shot</th> <th>3-shot</th> <th>0-shot</th> <th>1-shot</th> <th>3-shot</th> </tr> </thead> <tbody> <tr> <td rowspan="4">De</td> <td>bAbI-15</td> <td>✓</td> <td>85.00</td> <td>76.00</td> <td>75.00</td> <td>38.40</td> <td>46.40</td> <td>39.70</td> <td>79.00</td> <td>80.00</td> <td>88.00</td> <td>100 [39]</td> </tr> <tr> <td>EntailmentBank</td> <td>✓</td> <td>93.00</td> <td>88.00</td> <td>89.00</td> <td>83.82</td> <td>82.06</td> <td>77.94</td> <td>96.00</td> <td>97.00</td> <td>97.00</td> <td>100 [32]</td> </tr> <tr> <td>RuleTaker</td> <td></td> <td>64.00</td> <td>60.00</td> <td>62.00</td> <td>42.00</td> <td>38.00</td> <td>40.20</td> <td>64.00</td> <td>57.00</td> <td>70.00</td> <td>≈100 [40]</td> </tr> <tr> <td>FOLIO</td> <td></td> <td>48.00</td> <td>53.00</td> <td>52.00</td> <td>50.00</td> <td>50.98</td> <td>54.41</td> <td>52.00</td> <td>43.00</td> <td>49.00</td> <td>62.11 [33]</td> </tr> <tr> <td rowspan="2">In.</td> <td>Leap-Of-Thought</td> <td></td> <td>82.00</td> <td>90.00</td> <td>87.00</td> <td>72.61</td> <td>74.01</td> <td>61.21</td> <td>79.00</td> <td>72.00</td> <td>79.00</td> <td>99.7 [34]</td> </tr> <tr> <td>bAbI-16</td> <td>✓</td> <td>84.00</td> <td>81.00</td> <td>74.00</td> <td>17.10</td> <td>24.70</td> <td>12.90</td> <td>73.00</td> <td>44.00</td> <td>52.00</td> <td>100 [39]</td> </tr> <tr> <td rowspan="4">Ab.</td> <td>CLUTRR</td> <td>✓</td> <td>6.00</td> <td>23.00</td> <td>20.00</td> <td>21.99</td> <td>19.55</td> <td>12.83</td> <td>23.00</td> <td>26.00</td> <td>24.00</td> <td>95.0 [44]</td> </tr> <tr> <td>α-NLI</td> <td></td> <td>74.00</td> <td>70.00</td> <td>74.00</td> <td>80.90</td> <td>80.00</td> <td>79.10</td> <td>75.00</td> <td>74.00</td> <td>77.00</td> <td>68.90 [41]</td> </tr> <tr> <td>α-NLG</td> <td>✓</td> <td>9.00</td> <td>10.00</td> <td>12.00</td> <td>21.90</td> <td>23.40</td> <td>25.90</td> <td>10.00</td> <td>12.00</td> <td>15.00</td> <td>45.00 [41]</td> </tr> <tr> <td>AbductiveRules</td> <td>✓</td> <td>75.00</td> <td>42.00</td> <td>35.00</td> <td>23.30</td> <td>35.10</td> <td>29.80</td> <td>71.00</td> <td>49.00</td> <td>22.00</td> <td>100 [42]</td> </tr> <tr> <td rowspan="4">Mix</td> <td>D*Ab</td> <td>✓</td> <td>8.00</td> <td>21.00</td> <td>23.00</td> <td>11.60</td> <td>2.50</td> <td>1.80</td> <td>11.00</td> <td>0.00</td> <td>0.00</td> <td>≥95 [43]</td> </tr> <tr> <td>ReClor</td> <td></td> <td>53.00</td> <td>53.00</td> <td>55.00</td> <td>58.80</td> <td>56.00</td> <td>58.80</td> <td>56.00</td> <td>55.00</td> <td>56.00</td> <td>75.00 [45]</td> </tr> <tr> <td>LogiQA</td> <td></td> <td>41.00</td> <td>35.00</td> <td>39.00</td> <td>40.25</td> <td>39.48</td> <td>40.86</td> <td>48.00</td> <td>46.00</td> <td>47.00</td> <td>46.10 [45]</td> </tr> <tr> <td>LogiQA 2.0</td> <td></td> <td>43.00</td> <td>42.00</td> <td>41.00</td> <td>54.60</td> <td>50.80</td> <td>54.80</td> <td>53.00</td> <td>46.00</td> <td>47.00</td> <td>72.25 [38]</td> </tr> <tr> <td rowspan="2">Long</td> <td>LogiQA2NLI</td> <td></td> <td>59.00</td> <td>55.00</td> <td>58.00</td> <td>57.83</td> <td>53.83</td> <td>57.00</td> <td>48.00</td> <td>50.00</td> <td>47.00</td> <td>≈80</td> </tr> <tr> <td>Long</td> <td></td> </tr> </tbody> </table>		Dataset	Gen.	text-davinci-003			ChatGPT			BARD			SOTA	0-shot	1-shot	3-shot	0-shot	1-shot	3-shot	0-shot	1-shot	3-shot	De	bAbI-15	✓	85.00	76.00	75.00	38.40	46.40	39.70	79.00	80.00	88.00	100 [39]	EntailmentBank	✓	93.00	88.00	89.00	83.82	82.06	77.94	96.00	97.00	97.00	100 [32]	RuleTaker		64.00	60.00	62.00	42.00	38.00	40.20	64.00	57.00	70.00	≈100 [40]	FOLIO		48.00	53.00	52.00	50.00	50.98	54.41	52.00	43.00	49.00	62.11 [33]	In.	Leap-Of-Thought		82.00	90.00	87.00	72.61	74.01	61.21	79.00	72.00	79.00	99.7 [34]	bAbI-16	✓	84.00	81.00	74.00	17.10	24.70	12.90	73.00	44.00	52.00	100 [39]	Ab.	CLUTRR	✓	6.00	23.00	20.00	21.99	19.55	12.83	23.00	26.00	24.00	95.0 [44]	α-NLI		74.00	70.00	74.00	80.90	80.00	79.10	75.00	74.00	77.00	68.90 [41]	α-NLG	✓	9.00	10.00	12.00	21.90	23.40	25.90	10.00	12.00	15.00	45.00 [41]	AbductiveRules	✓	75.00	42.00	35.00	23.30	35.10	29.80	71.00	49.00	22.00	100 [42]	Mix	D*Ab	✓	8.00	21.00	23.00	11.60	2.50	1.80	11.00	0.00	0.00	≥95 [43]	ReClor		53.00	53.00	55.00	58.80	56.00	58.80	56.00	55.00	56.00	75.00 [45]	LogiQA		41.00	35.00	39.00	40.25	39.48	40.86	48.00	46.00	47.00	46.10 [45]	LogiQA 2.0		43.00	42.00	41.00	54.60	50.80	54.80	53.00	46.00	47.00	72.25 [38]	Long	LogiQA2NLI		59.00	55.00	58.00	57.83	53.83	57.00	48.00	50.00	47.00	≈80	Long											
	Dataset				Gen.	text-davinci-003			ChatGPT			BARD			SOTA																																																																																																																																																																																																													
		0-shot	1-shot	3-shot		0-shot	1-shot	3-shot	0-shot	1-shot	3-shot																																																																																																																																																																																																																	
De	bAbI-15	✓	85.00	76.00	75.00	38.40	46.40	39.70	79.00	80.00	88.00	100 [39]																																																																																																																																																																																																																
	EntailmentBank	✓	93.00	88.00	89.00	83.82	82.06	77.94	96.00	97.00	97.00	100 [32]																																																																																																																																																																																																																
	RuleTaker		64.00	60.00	62.00	42.00	38.00	40.20	64.00	57.00	70.00	≈100 [40]																																																																																																																																																																																																																
	FOLIO		48.00	53.00	52.00	50.00	50.98	54.41	52.00	43.00	49.00	62.11 [33]																																																																																																																																																																																																																
In.	Leap-Of-Thought		82.00	90.00	87.00	72.61	74.01	61.21	79.00	72.00	79.00	99.7 [34]																																																																																																																																																																																																																
	bAbI-16	✓	84.00	81.00	74.00	17.10	24.70	12.90	73.00	44.00	52.00	100 [39]																																																																																																																																																																																																																
Ab.	CLUTRR	✓	6.00	23.00	20.00	21.99	19.55	12.83	23.00	26.00	24.00	95.0 [44]																																																																																																																																																																																																																
	α-NLI		74.00	70.00	74.00	80.90	80.00	79.10	75.00	74.00	77.00	68.90 [41]																																																																																																																																																																																																																
	α-NLG	✓	9.00	10.00	12.00	21.90	23.40	25.90	10.00	12.00	15.00	45.00 [41]																																																																																																																																																																																																																
	AbductiveRules	✓	75.00	42.00	35.00	23.30	35.10	29.80	71.00	49.00	22.00	100 [42]																																																																																																																																																																																																																
Mix	D*Ab	✓	8.00	21.00	23.00	11.60	2.50	1.80	11.00	0.00	0.00	≥95 [43]																																																																																																																																																																																																																
	ReClor		53.00	53.00	55.00	58.80	56.00	58.80	56.00	55.00	56.00	75.00 [45]																																																																																																																																																																																																																
	LogiQA		41.00	35.00	39.00	40.25	39.48	40.86	48.00	46.00	47.00	46.10 [45]																																																																																																																																																																																																																
	LogiQA 2.0		43.00	42.00	41.00	54.60	50.80	54.80	53.00	46.00	47.00	72.25 [38]																																																																																																																																																																																																																
Long	LogiQA2NLI		59.00	55.00	58.00	57.83	53.83	57.00	48.00	50.00	47.00	≈80																																																																																																																																																																																																																
	Long																																																																																																																																																																																																																											
<h3>L-Eval: Instituting Standardized Evaluation For Long Context Language Models</h3>	<p>This article is dedicated to a comprehensive evaluation of long-text instructions, covering a variety of domains and task formats:</p>																																																																																																																																																																																																																											

Data-name	Instruction-style	#Samples	#Insts	Doc-len	Inst-len	Ans-len
coursera	multiple-choice question	15	172	6950	52	1
gsm100	school math	100	100	0	3335	1
quality	multiple-choice question	15	202	4161	58	12
topic_retrieval	retrieving topics	50	150	8843	17	6
tpo	multiple-choice question	15	269	2986	53	1
financial_qa	abstractive QA	6	52	4000	39	61
legal_contract_qa	abstractive QA	20	130	18115	36	64
multidoc_qa	abstractive QA	20	136	2802	14	21
natural_question	abstractive QA	20	104	13245	10	4
narrative_qa	abstractive QA	20	182	32805	10	10
scientific_qa	abstractive QA	20	160	3238	7	12
gov_report_summ	summarization	13	13	4420	7	264
meeting_summ	query-based summarization	20	156	11441	10	60
news_summ	summarization	11	11	4658	5	274
paper_assistant	completing and reviewing papers	20	60	6145	76	275
patent_summ	summarization	13	13	4025	12	124
review_summ	query-based summarization	20	120	14789	14	97
tv_show_summ	summarization	13	13	5834	6	98

The conclusion is as follows:

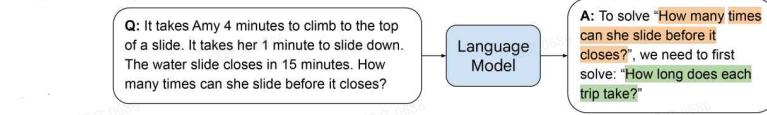
- Using LLM for automated evaluation of long texts is difficult, but GPT-4 works well, but it is expensive
- The gap between open-source and business models is significant
- Long contexts don't always lead to performance improvements
- Retrieval-based solutions cannot solve all long document modeling tasks (some tasks require comprehensive processing of information in context).

Least-To-Most Prompting Enables Complex Reasoning In Large Language Models

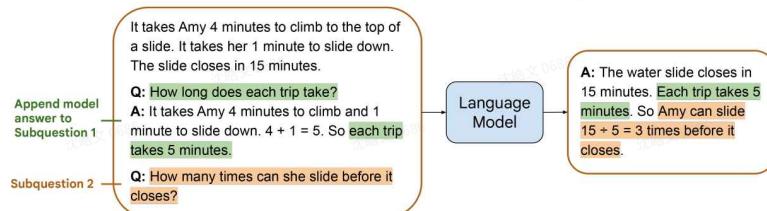
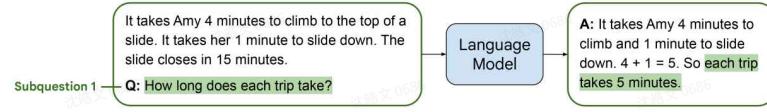
Contribution point 1 of this article:

Unlike the CoT method, this paper proposes a way to construct prompts called Last-to-Most, and emphasizes two stages of solving complex problems: (1) task decomposition; (2) task resolution, where task resolution relies primarily on the output of previously solved subtasks, i.e. the current state.

Stage 1: Decompose Question into Subquestions



Stage 2: Sequentially Solve Subquestions



To test the Last-to-most prompting strategy proposed in this paper

The contribution point 2 of this article tests the ability of LLM in complex logical reasoning using the following three benchmarks.

- String operation: Constructed a test set of string operations. The purpose of the task is to extract and concatenate the last letter of each word in a word list. For example, ["listen", "read", "write "] -> 'n de '.
- Experimental results on GSM8K show that Best-to-most prompting is significantly better than CoT in processing complex tasks.
- The dataset SCAN for verifying combinatorial generalization ability tests the generalization ability of the model by mapping natural language to corresponding action sequences.

Command	Action Sequence
"look thrice after jump"	JUMP LOOK LOOK LOOK
"run left and walk"	TURN_LEFT RUN WALK
"look opposite right"	TURN_RIGHT TURN.RIGHT LOOK

The difficulty of this dataset lies in the fact that the samples in the training data are significantly simpler than the test samples (including shorter action sequences and simpler patterns). The method of testing this type of test model from simple exemplars to more complex problems is usually called the combinatorial generalization problem. Similar datasets or benchmarks can effectively test whether the model has learned the ability to reason rather than simply memorizing the dataset.

Continuing from the Left-to-most paper, this paper first proposes a Prompting strategy for decomposing the Modularization task. Different decomposition tasks can be solved through specific models, Prompting LLM, or symbolic tools.

The benchmark for testing is as follows:

- Extract the kth letter of each word in the list and concatenate it (string operation).
- Sequence flip ability test
- CommaQA-E dataset (Long-Context QA): Issues that require multi-hop understanding
- Open-Domain QA (multi-hop + API call): Combining multi-hop inference to query the test data generated by the API
 - 2Wiki-MultihopQA
 - MuSiQue
 - HotpotQA

Decomposed Prompting: A MODULAR APPROACH FOR SOLVING COMPLEX TASKS

TULLE: Towards a Useful Chinese benchmark for Large Language modEls

We plan to propose a dataset for complex Chinese instructions,

	<p>which includes the following complex logical reasoning content.</p> <ul style="list-style-type: none"> ◦ Summary of Long Text Keywords ◦ Closed Q & A: Often long text Q & A ◦ Long text summary ◦ Complex format extraction 	<p>从句子中提取公司的基本信息。需要提取公司名称、成立时间、公司类型、资产总额、经营总收入、主要销售区域、主要产品等信息。其中的输出格式为输出格式为字典，包含以下键值对：</p> <pre>{ "公司名称": "", "成立时间": "", "公司类型": "", "资产总额": "", "经营总收入": "", "主要销售区域": "", "主要产品": "" }</pre>
<h3>ALERT: Adapting Language Models to Reasoning Tasks</h3>	<p>LLM is evaluated based on commonly used logical reasoning abilities, comprehensively evaluated on 20 datasets of 10 reasoning tasks, including: logic, causality, common sense, induction, space, analogy, deduction, argument, textual implication, and mathematical relevance.</p> <p>From the observation of experimental results, the tasks that LLM currently finds difficult to handle include: common sense reasoning, mathematical reasoning: mathematical logic + symbolic manipulation, and spatial reasoning. Other experimental conclusions are as follows:</p> <ul style="list-style-type: none"> • The reasoning ability of LLM is not obtained by memorizing training data • Fine-tune can improve some reasoning abilities, but not all (common sense reasoning). • Fine-tune will make LLM overfit templates and affect the generalization of its reasoning ability 	
<p>https://www.refuel.ai/blog-posts/llm-labeling-technical-report</p>	<p>This article is a technical report that provides a comprehensive evaluation result of LLM's ability to annotate text (measuring annotation quality, practice, cost), and makes the following findings:</p> <ul style="list-style-type: none"> • The effect of LLM completing Data Annotation is not much different from that of human, but it is 20 times faster and 7 times cheaper • GPT-4 has the best annotation effect, but if you need to balance your income and expenses, GPT-3.5-turbo, PaLM-2, FLAN-T5-XXL are also okay • Threshold-based Confidence Level estimation is an effective method to improve labeling quality, reduce labeling errors and illusions. Specifically, using token-level log probability averaging can be an effective method for evaluating Confidence Level. The stricter the Confidence Level requirements, the higher the agreement • In-Context learning and CoT can improve annotation quality 	

Dataset	Task	Description
Banking77	Classification (multi-class)	Item classification of online banking queries
LEDGAR	Classification (multi-class)	Contract provision classification for US SEC filings
Civil comments	Classification (binary)	Toxicity detection in public user comments
WalmartAmazon	Entity matching (de-duplication)	Entity matching task from Walmart and Amazon
Company Match	Entity matching (de-duplication)	Company descriptions from Wikipedia and company websites
SQuAD	Question Answering	Science exam questions (Physics, Chemistry and Biology)
SQuAD 2.0	Question Answering	Wikipedia reading comprehension dataset
CoNLL	Named Entity Recognition	NER dataset consisting of news articles from the Reuters corpus

Provider	Model	Dataset								Average
		Banking77	Civil Comments	LEDGAR	Walmart Amazon	Company match	SQuAD2.0	SciQ	CoNLL 2003	
	gpt-4	0.843	0.88	0.78	0.98	0.965	0.801	0.969	0.856	0.884
	gpt-3.5-turbo	0.776	0.785	0.734	0.962	0.85	0.69	0.944	0.762	0.813
	claude-v1	0.828	0.854	0.715	0.922	0.7	0.759	0.955	0.746	0.81
	text-davinci-003	0.806	0.89	0.673	0.777	0.72	0.586	0.913	0.774	0.767
	text-bison@001	0.775	0.87	0.588	0.949	0.946	0.781	0.947	0.623	0.806
	flan-t5-xxl	0.792	0.88	0.713	0.951	0.805	0.818	0.839	0.419	0.777
	Human annotators	0.77	-	-	0.966	-	-	-	0.849	0.862

Label quality (% agreement with ground truth labels) across a variety of NLP tasks. Highest numbers in each column are highlighted in green

However, some issues with this job are quite obvious.

- Lack of evaluation of more complex datasets and tasks (text generation)
- Lack to evaluate more test domains

Can foundation models label data like humans?
(<https://hf.co/blog/llm-leaderboard>)

This article is a summary of Huggingface's specific practical work on LLM evaluation. **It analyzes the ability of LLM evaluation text generation results**. The main conclusion of this article is "**The evaluation system based on LLM needs to be carefully selected and used**". Through a large amount of work related to consistency evaluation and manual evaluation, the following conclusions are drawn in this article:

- There is a clear position bias in LLM-based evaluations, that is, LLM tends to be the first to respond
- Based on LLM evaluation, especially GPT-4, there is a clear length bias, that is, the more content generated, the higher the score

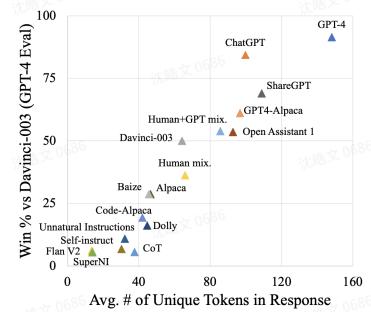
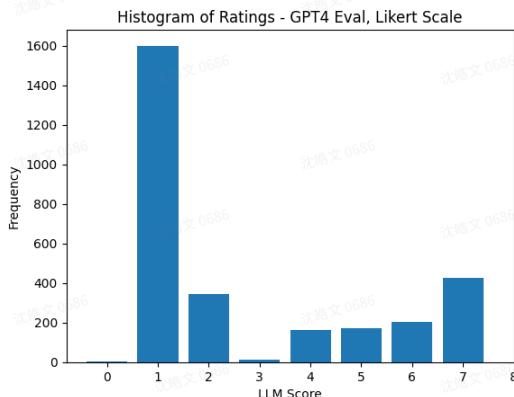


Figure 2: Win-rate scores of 13B models (trained on different datasets) given by GPT-4 strongly correlate with the average numbers of unique tokens in the model responses (Pearson $r = 0.96$).

- Compared with manual evaluation, GPT-4 scores high in creative content generation and human relevance, but poor in factual content generation (especially code content related evaluation) and human relevance. In terms of task type, the correlation between humans and GPT-4 is highest in "high entropy" tasks (such as brainstorming and generation tasks), while the correlation is low in programming tasks. (This indicates that automatic code evaluation is still difficult and requires considerable accuracy).

Category	Correlation: GPT-4 to Human Labels
Brainstorm	0.60
Creative generation	0.55
Commonsense reasoning	0.46
Question answering	0.44
Summarization	0.40
Natural language to code	0.33

- Directly generating scores for each reply instead of comparing multiple replies can reduce position bias
- In the Prompt, it is explicitly required that GPT-4 should not retain this position bias, which will cause the model to lean to the other side, but not as bad as the above situation.

A Hazard Analysis Framework for Code Synthesis Large Language Models

This article designs an evaluation scheme for the ability of OpenAI Codex system to generate code for prompt information for complex and high-level problems.

- Abstract norms
 - High-level abstraction (the purpose of the system)
 - Low-level abstraction (related to specific implementation details)
- Inference of computational state
 - Variable relationships: tracking variables, variable relationships
 - Complexity of restrictions: ensuring that there are no error states and running towards the correct state
- Concurrency
- Randomness
- Other capabilities
 - The ability to reuse code and parameters
 - Automatically determine the structure of the program: subroutines, loops, recursion, internal storage variables, input variables, etc
 - Programming to build applications: library functions, pointers and other programming application skills
 - Clear definition
 - Wide range of applications

Analysis found some factors that affect the generation ability of Codex:

1. Existing code context provided by the user
2. Defined function and variable names
3. Existing comments and documentation

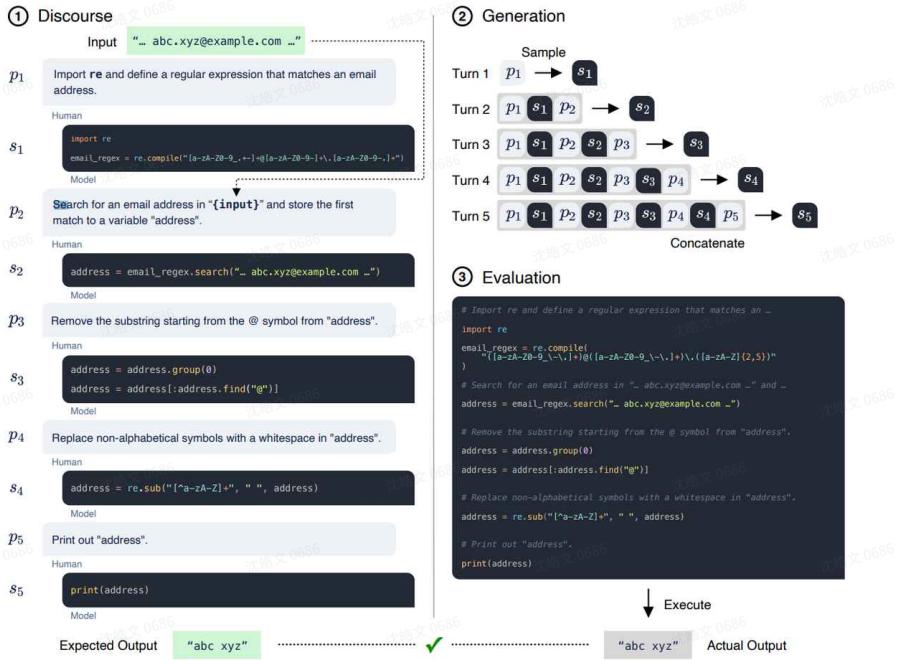
4. Distribution of training data
5. Length and conciseness of prompt

The conclusion is as follows:

- Relationship between variables:
Codex can do good inference on 2-3 variables (input and output variables), but it is difficult to do more than 3
- Limit complexity:
For short and specific requirements (such as avoiding division by zero errors, when to exit the loop, etc.), Codex answers well; but for more complex and rare restrictions, it is difficult to handle, such as using macro instructions, library functions, typing, pointers, conditional operations, typed functions, etc
- Concurrency is poor and cannot generate correct fair, atomic, and synchronous code
- Non-deterministic is good for simple random number generation and Machine Learning code construction, but ML code is only good for common content generation (such as MNIST), but it can be used as a template builder for ML code.
- High-level constraints and the ability to automatically determine program structure.
Codex has good ability in building a function or a module, but its ability to automatically determine multiple modules to solve problems is poor
- Poor understanding and processing ability of long prompts may be related to training data

Codegen: An open large language model for code with multi-turn program synthesis

In addition to proposing a code generative model, this article also proposes an evaluation scheme for Multi-turn program synthesis (MTPB). Specifically, the MTPB evaluation benchmark tests the model's ability in two aspects: (1) generating code based on the current description; (2) generating code using functions or variables defined in previous steps.

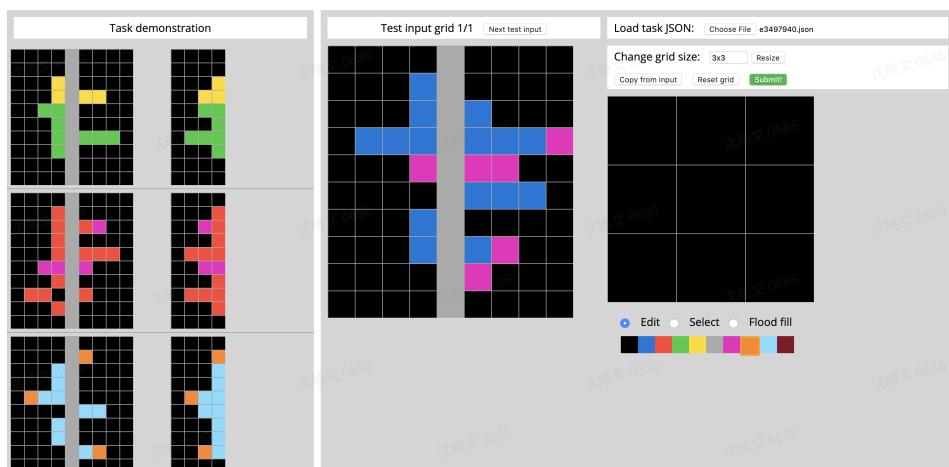


Abstraction and Reasoning Challenge (ARC Corpus)

AND

The ConceptARC Benchmark: Evaluating Understanding and Generalization in the ARC Domain

In order to test the ability of existing AI to solve complex problems with a limited number of samples, this paper constructs a very complex inductive inference dataset ARC, and accurately generates the output results on the test set by analyzing the display examples in three training sets:



However, the problem is too complex. The ConceptARC dataset simplifies the ARC dataset, summarizes some inference elements, and conducts a comprehensive analysis of GPT-4, resulting in the following conclusion:

Concept	Humans	ARC-Kaggle First Place	ARC-Kaggle Second Place	GPT-4
Above and Below	0.90	0.70	0.33	0.23
Center	0.94	0.50	0.20	0.33
Clean Up	0.97	0.50	0.20	0.20
Complete Shape	0.85	0.47	0.30	0.23
Copy	0.94	0.23	0.27	0.23
Count	0.88	0.60	0.40	0.13
Extend To Boundary	0.93	0.77	0.47	0.07
Extract Objects	0.86	0.43	0.43	0.03
Filled and Not Filled	0.96	0.73	0.43	0.17
Horizontal and Vertical	0.91	0.43	0.10	0.27
Inside and Outside	0.91	0.57	0.10	0.10
Move To Boundary	0.91	0.37	0.30	0.20
Order	0.83	0.27	0.23	0.27
Same and Different	0.88	0.53	0.17	0.17
Top and Bottom 2D	0.95	0.60	0.57	0.23
Top and Bottom 3D	0.93	0.50	0.03	0.20

The experimental results show that GPT-4 still lags significantly behind real human performance, indicating that its abstract understanding ability still needs to be improved

2.2 Virtual scene related

Most of the evaluation plans and application scenarios for LLM mentioned above are limited to the pure text field. Currently, researchers are also focusing on comprehensively evaluating the abstract understanding ability by combining LLM and virtual environment fields. Among them, the tasks in this virtual environment mainly involve the following three types:

2.2.1 virtual world

This article designs a virtual space for LLM agents to test LLM's ability to solve complex problems by testing the scientific experiments they complete in it. In order to simplify the evaluation process of this environment, ScientWorld decomposes complex scientific experiment-related tasks into multiple sub-tasks to complete one by one, and comprehensively evaluates LLM's ability to complete a complex task by checking the status of LLM in each sub-task.



ScienceWorld: Is your Agent Smarter than a 5th Grader?

SCIENCEWORLD Task 3 (Test Electrical Conductivity Subtask)

Task Description

Your task is to determine if a metal fork is electrically conductive. The metal fork is located around the kitchen. First, focus on the metal fork. If conductive, place it in the blue box. If nonconductive, place it in the green box.

Agent starts in a random location, navigates to kitchen

Kitchen

This room is called the kitchen. In it, you see: a fridge. The freezer door is closed. a cupboard. The cupboard door is closed. a table. On the table is: a glass cup (containing nothing). a stove, which is turned off. On the stove is: nothing. a sink, which is turned off. In the sink is: nothing. a thermometer, currently reading a temperature of 10 degrees celsius a metal fork a counter. On the counter is: a drawer, a bowl (containing an orange, a banana, a potato, a red apple). You also see:
A door to the outside (that is open)
A door to the bathroom (that is open)
A door to the hallway (that is open)

Action: **focus on metal fork**

You focus on the metal fork.

Action: **pick up metal fork**

You move the metal fork to the inventory.

In room with task object → **0.1**

SCORE
0.5

Agent navigates to room with electrical components

Workshop

This room is called the workshop. In it, you see: a electric buzzer, which is off a electric motor, which is off a solar panel, which is off a table. On the table is: a blue wire, a battery, a red light bulb, which is off, a black wire, a switch, which is off, a orange wire.

a blue answer box

a green answer box

You also see:

A door to the hallway (that is open)

Action: **connect battery cathode to orange wire terminal 1**
cathode on battery is now connected to terminal 1 on orange wire.

Action: **connect metal fork terminal 1 to cathode in red light bulb**
terminal 1 on metal fork is now connected to cathode on red light bulb.

Agent continues wiring a circuit containing the metal fork

Action: **examine red light bulb**
a red light bulb, which is on.

Action: **move metal fork to blue answer box**
(disconnecting metal fork) You move the metal fork to the blue box.

Task Completed.

In room with components → **0.6**

SCORE
0.6

Successfully made circuit

SCORE
0.7

SCORE
1.0

2.2.2 Network Development

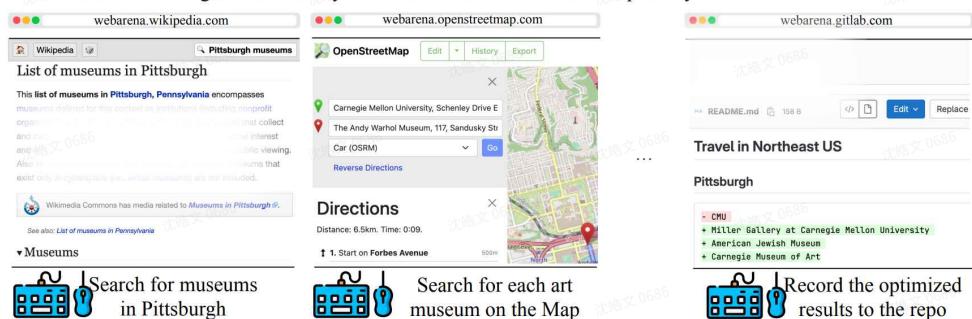
WebArena: A Realistic Web Environment for Building Autonomous Agents

This article proposes a comprehensive and realistic benchmark for testing LLM's ability to complete tasks and use tools related to network search. By simulating the behavior of real users searching the network, the user's online search tasks are roughly classified.

- E-commerce shopping
- Social networking platform applications (Reddit Facebook...)
- Crowdsourced software development (GitLab)
- CMS (such as online shopping website management)

Based on this, some commonly used tools (calculators, navigation software, etc.) and common knowledge data (such as WikiLingo) are provided. LLM is required to complete specific tasks according to the user's input intention, as shown in the following figure.

 “ Create an efficient itinerary to visit all Pittsburgh's art museums with minimal driving distance starting from CMU. Log the order in my “awesome-northeast-us-travel” repository ”



Different evaluation schemes are adopted for different user intentions, as shown below:

- For the task of Information seeking, exact match, must_include, fuzzy_match are used to determine the generation of replies and standard replies
- For navigation or content configuration related tasks, compare and judge the state after LLM performs an action with the ground-truth state

.....

The experimental results show that the existing LLM has poor ability to complete existing tasks, even when using techniques similar to CoT.

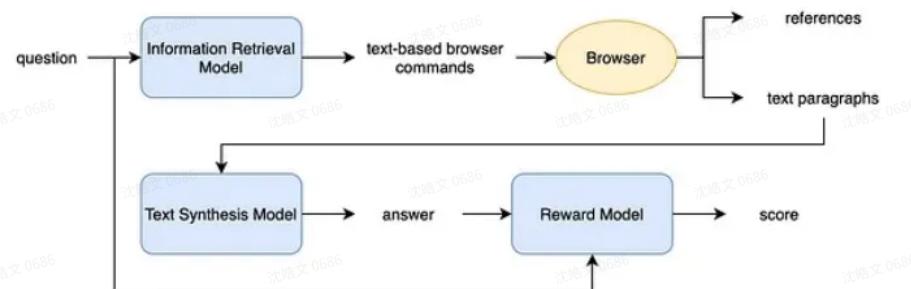
Agent	Model	Success Rate
Direct	gpt-3.5	7.14
Reasoning	gpt-3.5	7.38
Reasoning	gpt-4	10.59

By analyzing the types of errors, it was found that the current points where LLM is prone to errors are:

- Premature termination of action sequence (uncertain infeasible state determination): For tasks that require a series of actions to complete, LLM sometimes directly terminates the task prematurely, such as "directly forking all repositories of xxx from Git", but LLM often directly reports that no repositories have been found, thus directly terminating the task
- Observation bias: In tasks that require comprehensive comparison of multiple information to make a final response, LLM often directly selects top-1 results to return, ignoring other feasible content
- Incorrect understanding of the state: LLM often mistakenly summarizes the current state, such as ignoring some fine grain information in longer texts, resulting in task failure

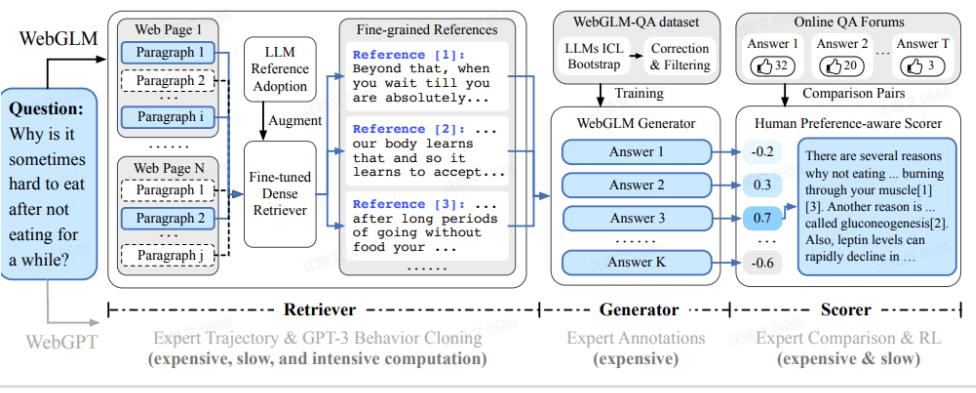
WebGPT/WebCPM

WebGPT solves long-text question answering tasks (LFQA tasks, benchmarks usually use TriviaQA) through search. By combining text searchers, LLM, and reward models, using human feedback methods, the model is trained to search the network, select useful information, and then generate factually correct reply content. The overall architecture diagram is as follows:



WebCPM and WebGLM have similar principles. In contrast, WebGLM reduces the training cost of Web-based LLM through the following three techniques.

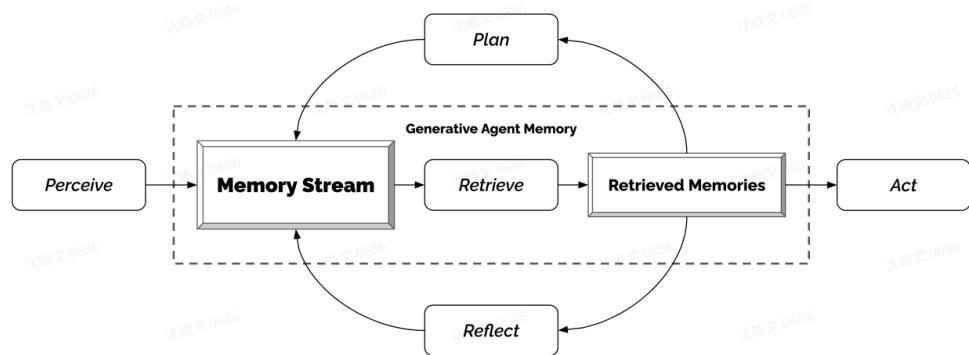
- Using LLM method to generate reference data distillation and refining model screening reference
- Build generated data with reference information from the reference generation dataset in the previous step using in-context learning
- Train the reward model of human preference through online scoring data of social networking sites



2.2.3 Crowdsourcing

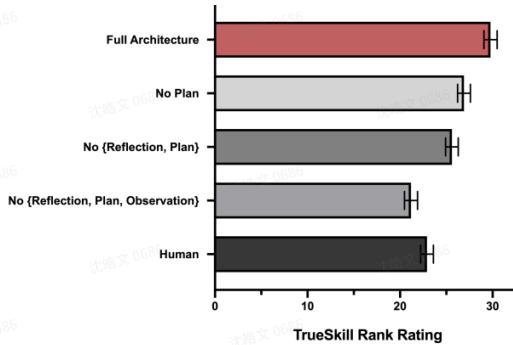
Generative Agents: Interactive Simulacra of Human Behavior

This article studies generative agents, using generative models to simulate real human social activities in daily life. The author also designs a new agent architecture that includes important operational modules such as observation, planning, and reflection.



In this open virtual environment, the author first uses controlled evaluation (manual evaluation) to conduct a survey similar to an "interview" on the agent to evaluate its credibility. Specifically, each annotator will ask the agent five types of questions. Based on the annotator's judgment of the current simulated situation, these five types of questions are:

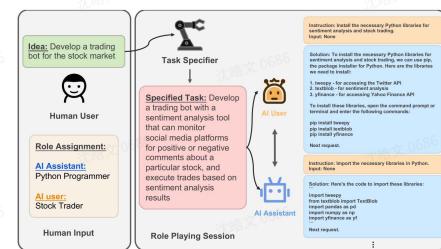
- Inquire about information related to the agent itself and observe whether the agent has a good understanding of itself
- Memory: Determine the ability of the agent to store memory by retrieving the relevant experiences and tasks of the agent
- Plans: Check the predictability of the agent's own plan and verify the credibility of each agent
- Reaction: By analyzing the feedback made by the agent's observation of external states, determine whether it truly understands the impact of past events and its own information
- Reflection: Let the agent reflect and check their ability to deeply understand other people in the community and what is happening



The experimental results verify the effectiveness of the architecture constructed by the current author. In addition, the author conducted a complex end-to-end evaluation to analyze the stability and social ability of the agent, which will not be elaborated in detail.

This article contributed multiple datasets to the task of "AI Collaboration", covering fields such as AI Society, Code, Math, etc.

Data (Hosted on Hugging Face)			
Dataset	Chat format	Instruction format	Chat format (translated)
AI Society	Chat format	Instruction format	Chat format (translated)
Code	Chat format	Instruction format	x
Math	Chat format	x	x
Physics	Chat format	x	x
Chemistry	Chat format	x	x
Biology	Chat format	x	x



By designing a role-playing framework, multiple AI agents can cooperate to complete tasks without human supervision (which is often very time-consuming and expensive). The framework mainly includes three parts: task specific agent, AI-user agent, and AI-assistant agent.

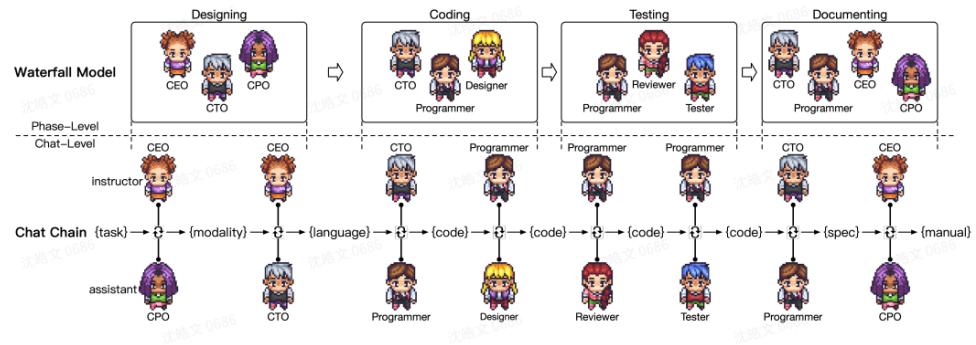
AI-assistant is responsible for depicting the user's ideas and intentions into more accurate requirements, and then AI-user replaces manual guidance to AI-assistant to complete the task according to the generated intentions. During the experiment, the author found that current LLM often make the following errors when solving this collaboration problem.

- Role flipping: AI-assistant and AI-User often flip roles, which needs to be carefully designed through prompts to solve
- Assistant repeats instructions: AI-assistant often repeats user instructions without providing solutions
- Flake replies: AI-assistant often generates irrelevant replies that do not help solve the current instruction
- Infinite loop: AI-assistant and AI-user sometimes get stuck in endless looping conversations like "thank you" and "bye" and cannot escape.

CAMEL: Communicative Agents for “Mind” Exploration of Large Scale Language Model Society

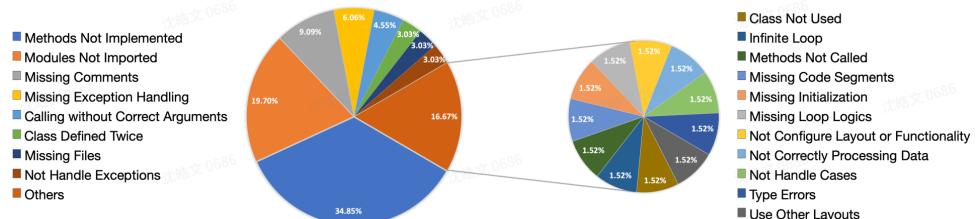
Communicative Agents for Software Development (ChatDev)

This article draws on the waterfall model of software development and constructs a virtual software development team to solve users' project-level code requirements tasks. Similar to the previous paper, this article divides the team into the following parts according to responsibilities.

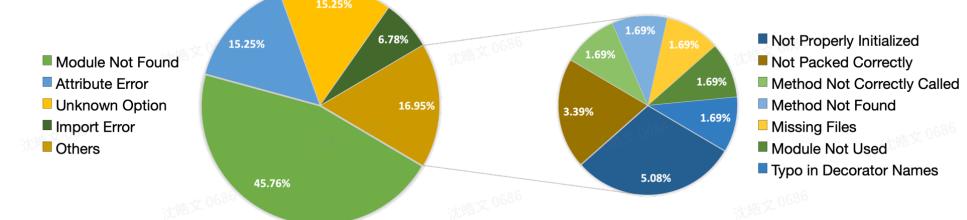


By **collecting and organizing 70 code-built test projects on the Camel dataset**, we tested the ability of ChatGPT (20 languages, 50 domains, 50 tasks). The test results showed that the code built with ChatGPT ran perfectly in 86.66% of the test cases. 50% of the errors were caused by token length limitations, and the remaining errors were mainly caused by environment dependencies.

By examining the conversation between reviewer and coder, the following common issues were found in LLM:



By examining the dialogue between tester and coder, it was found that the common issues with LLM are as follows:



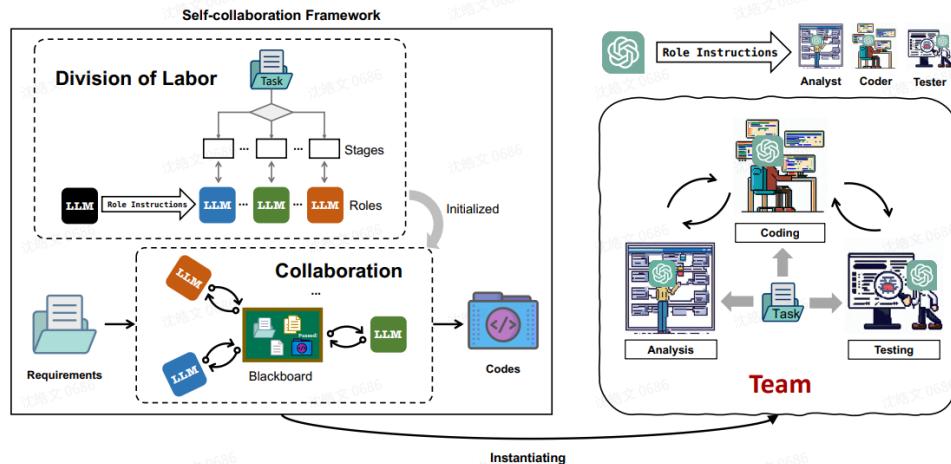
A large number of module not found errors and Methods Not Implemented (Modules Not Imported) errors indicate that LLM tends to overlook implementation details when building project-level code.

Self-collaboration Code Generation via ChatGPT

Unlike the task decomposition mentioned above, this article decomposes from the perspective of roles. By creating a virtual code team consisting of three people, including analysts, programmers, and testers, complex problems can be solved.

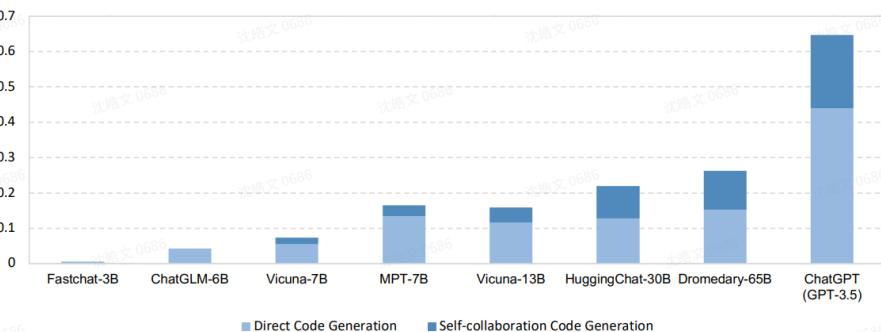
- Analysis is responsible for breaking down the problem
- Coding is responsible for encoding different sub-problems and building solutions

- Testing generates analysis reports for Coding responses (in natural language form, without performing unit tests) to provide feedback for updating code for the Coding module



After task decomposition, complex tasks are solved through cooperation among three roles, and a shared message queue called blackborad is used to synchronize their progress. The experimental conclusion is as follows:

- Using this role-playing-based crowdsourcing method can consistently improve the coding capabilities of various LLMs



- As the number of interactions gradually increases or decreases, the quality of the code increases (pass rate), but the cost also increases

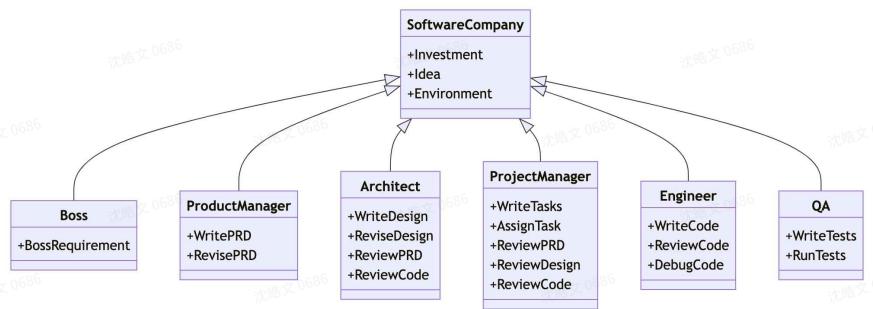
MI	HumanEval	HumanEval-ET	MBPP	MBPP-ET
0	45.1	35.3	47.5	34.3
1	60.4	50.7	53.0	38.0
2	62.2	50.7	54.1	38.5
4	63.5	51.9	55.5	40.8

- Crowdsourcing division of labor can significantly improve code quality (introducing AI-assistant's critique and task decomposition capabilities).

MetaGPT

In addition, the recent hot project MetaGPT breaks down different responsibilities in software development into different execution units according to the logic of building complex software, and utilizes the collaboration between various LLMs to complete the development of complex tasks. QA and Engineer are similar to the Coder and Tester

components in the above paper, and complex collaboration completes the development, feedback, and update of code.



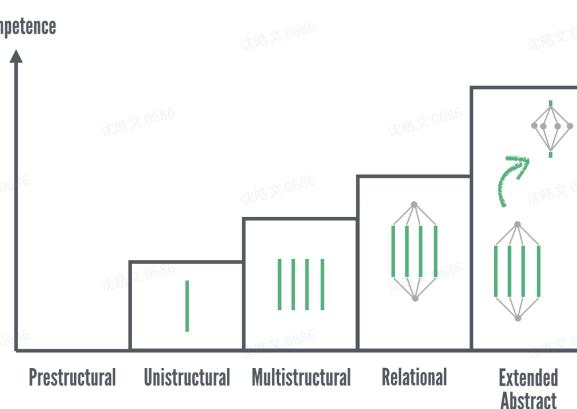
It can be found that researchers currently consider code analysis, evaluation, and correction to be a relatively complex task. A commonly used solution is to collaborate on code development, debugging, and testing by separating Coder and Tester.

3. Summarize from the perspective of cognitive theory

Is there any way to help us comprehensively sort out the complexity of the task from the perspective of cognition or education?

SOLO classification

A method to understand the process of learning becoming increasingly complex from the perspective of learning outcomes (outputs). Based on the conclusions of the outputs, the learning process can be divided into five stages from simple to complex.



1. Prestructural unstructured state : shallow understanding of ideas and problems, unable to effectively organize input structural information, or even completely do not understand ideas and problems

- Guessing about the problem
- Solve the problem in a very simple or even completely unrelated way

- 2. Unistructural state** : relatively one-sided and overly simplified understanding of ideas and problems, lack of comprehensive understanding of the significance and value of the problem, and some related keyword information is as follows
- 3. Multistructural state** : The understanding of the problem can be extended to multiple aspects, but the relationship between different aspects cannot be effectively organized
- 4. Relational organizational relationship state** : Understanding different aspects of a problem can be linked to see the overall logic and state behind the problem
- 5. Extended Abstract State** : Understanding and generalizing the information already mastered, and then applying it to new tasks or problems

From the above information, it seems that some clues of complex characteristics can be glimpsed, such as:

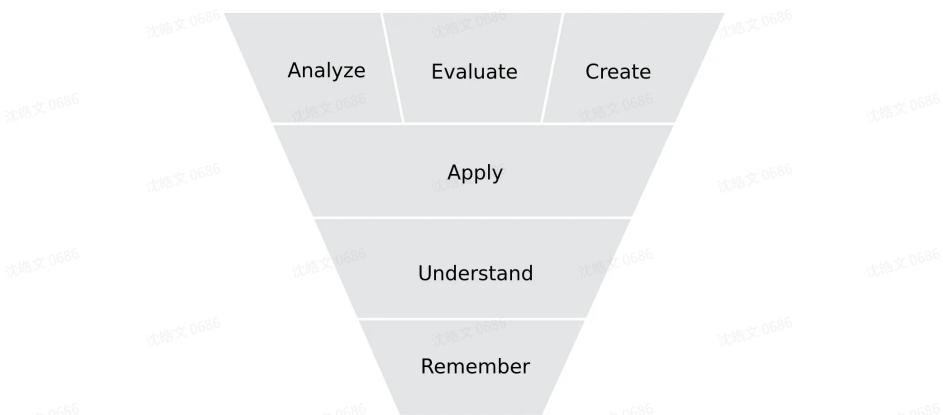
- The problem contains information from multiple dimensions (multivariate, multi-module, multi-angle, multi-step reasoning).
- There is an organized relationship between multiple different dimensions of the problem
- The ability to apply one example to another

Bloom Classification of Education

Educators can classify the process of learning and education according to three dimensions: "attitude", "skills", and "cognition". Among them, the "cognition" dimension is particularly important, including the following cognitive dimensions from simple to rare:

- Remember: to acknowledge or remember facts, terms, basic concepts, or answers without understanding their meanings
- Understand: Explain the main ideas and concepts and express their meaning through explanation, classification, summary, inference, comparison, and explanation.
- Application: Solve problems in new situations by applying knowledge, facts, techniques, and rules obtained. Use knowledge to solve problems, determine the connections between things, and their application in new situations.
- Analyze: **Examine information and break it down into components, determine the relationship between each part (the logical reasoning ability often mentioned in LLM)**, determine motivation or reasons, make inferences, and find evidence to support generalizations. Including: elemental analysis, relationship analysis, organizational analysis.

- Evaluation: Evaluation involves presenting and defending opinions by judging the validity or quality of information, ideas, or work based on a set of criteria. Its characteristics include: judgments on internal evidence, judgments on external standards. The main key words of information may include, **Evaluation, demonstration, proof, assessment, testing, judgment, rating, measurement, identification, selection, inspection, defense, determination, support, maintenance, criticism, comment, measurement, evaluation, selection, comparison, contrast, decision, estimation, rating, ratio, modification, rating, coordination, debate, deduction, induction, recommendation, monitoring, drawing conclusions, distinguishing, explaining (why), explaining, relationship, summarizing the problem, challenge, advocacy, persuasion**
- Create: involves putting elements together to form a coherent or functionally complete whole. Creation involves planning to reorganize elements into new styles or structures. This is the highest and highest level of the Bloom classification. Possible keywords include **design, combination, organization, reasoning, speculation, formulation, creation, creating new knowledge and information**



Starting from the cognitive dimension of learning and education from simplicity to difficulty, we can find that the difficulty of learning gradually increases with the transition from rote memorization, understanding, judgment, design and creation.

Slow and Fast Thinking

In Kahneman's **book "Thinking, fast and slow"**, the thinking process is divided into two types of systems, namely

- Fast thinking: Fast thinking is an automatic, intuitive, and effortless way of thinking, but it is also prone to errors and biases
- Slow thinking: Slow thinking is a controlled, rational, and hardworking way of thinking, but it is also more accurate and reliable

Among them, Kahneman believes that human thinking is a mixture of fast thinking and slow thinking. When we should use slow thinking, we often rely on fast thinking, and vice versa. LeCun and other researchers believe that LLM can be understood as fast thinking, but LLM lacks elements of

slow thinking (such as memory, logical reasoning, maintenance of internal states, prediction of environmental states, prediction of rewards, etc.), which may lead to errors, biases, and illusions in some tasks.

Some common examples of slow and fast thinking in humans summarized by Kahneman are as follows:

1. Fast think:

- a. Complete phrase
- b. Solve $2+2/1+1$
- c. Understand simple sentences
- d. Linking specific descriptions to specific things, such as fire and warmth
- e.

2. Slow think

- a. Record the number of letters A in a piece of text. [Maintain intermediate storage state and memory]
- b. Complete a task under certain constraints
- c. Determine the validity of a complex logical expression
- d. Trying to identify a certain sound
- e. Solve 17×24
- f.

Through the above cognitive methods and existing papers on the application and evaluation of LLM, we can roughly summarize the common characteristics of complex tasks.

- Having multiple tasks and logical relationships between tasks: **From the SOLO taxonomy, it can be found that high-level complex problems often have multiple aspects, and there are obvious logical relationships in the organization. This problem involves two aspects: task decomposition and task completion.**
 - Crowdsourced tasks in AI task specifier, assistant, user
 - Task sequence in Webarean
 - Waterfall model and internal dialogue designed in ChatDev
 - In common complex logical reasoning tasks, CoT is usually used to break down complex problems and organize them through linear sequences
 - Planning tasks: Starting from the existing state, plan ahead for future actions or sub-tasks to ensure the ability to complete the task correctly.

- Rules already have tasks, plan the sequence of completing tasks, such as the SwiftSage method in ScienceWorld, which mentions planning actions
 - Text-generated planning tasks mentioned in the GPT-4 report: local and global planning
 - How to plan the main function list of project-level code, etc
- Evaluation and analysis problems: **From the Bloom classification method, it can be found that using existing knowledge to complete judgment tasks and providing opinions is a high-level and more complex task**
 - Use LLM to complete automatic assessment, such as GPT-4 as judge
 - Use LLM to complete automatic critique of code, generate feedback, self-correction, etc
 - Additional memory and storage modules are required to assist in logical reasoning
 - Convert memory into symbolic database in ChatDB
 - New prompt techniques such as CoT/RAP store memory in the previous prompt to alleviate the defect of LLM's inability to build memory modules (the GPT-4 report also mentioned that the lack of additional memory modules for planning is an architectural problem of autoregressive models).
 - A large number of mathematical symbol reasoning, code execution prediction, code analysis and other tasks
 - The ability to apply existing knowledge to solve new problems
 - The GPT-4 report mentions the extensive use of library functions to complete tasks such as data lake visualization, deep learning code construction, and reverse engineering, and the ability to comprehensively utilize the knowledge already mastered in the pre-training stage (which may also involve knowledge through prompts or in-context learning, such as new tool functions) to solve new problems
 - In coding tasks, use existing programming skills to solve more complex programming problems
 - LeetCode
 - Project-level code build

4. Benchmark scenes and details

For the construction scenario of project-level code, the given overall project input and output:

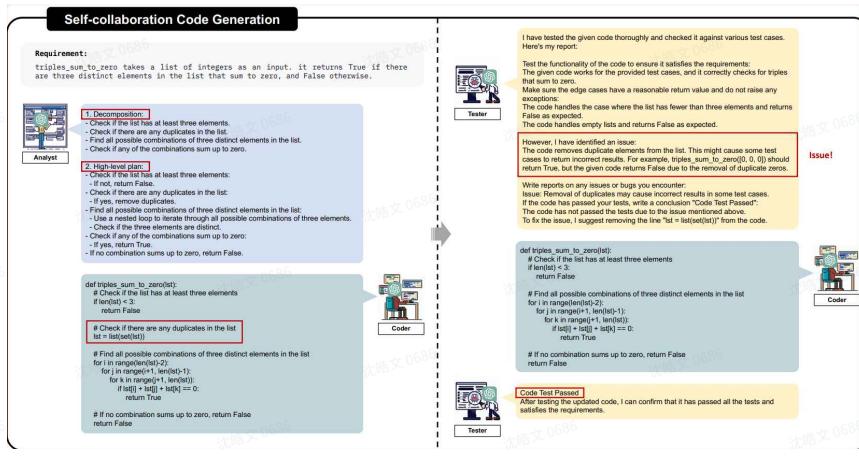
Input: Description of project-related requirements and some content details

Output: Complete details of the project

1. Design: including project planning (design proposal)

Camel's dataset only provides natural language descriptions of tasks, without clear definitions and constraints for different modules, which are continuously clarified through multiple rounds of interaction.

Self-collaboration Code Generation via ChatGPT seems to be able to do this part, **but there is no ability to systematically evaluate task planning, and it is not detailed enough, such as how the relationship between different modules is**



- a. What language is required (optional)
- b. What subfunctions or classes are needed (task splitting and planning)?
- c. Design of specific relationships between different sub-functions (linear or DAG).

2. Coding and testing

a. Coding

[Optional] There is currently no evaluation of the quality of the generated document (and whether the document quality needs to be evaluated).

Evaluation of generating unit test files (this is not available, and there is currently no work involved in generating unit tests based on task descriptions).

- What is the significance of automatically generating unit tests? Remove the degree of human involvement and automate as much as possible
 - i. The specific implementation of each subfile (including mutual calls between them).
 1. And detailed documentation for each sub-file (code explanation and documentation)
 2. Explanation information, configuration information, dependent library functions or dependent files for each unit
 - ii. Use existing class or module information to complete existing current functions (the ability to apply existing tools to solve new tasks).
 1. Recall existing code files and documents as additional input

iii. Generate unit test examples

b. Test

Improving Code Generation by Training with Natural Language Feedback has some of this data (natural language feedback), but the feedback and refinement data in this article need to be manually annotated

- Automatic evaluation scheme for natural language feedback
- Automatic evaluation scheme for code correction

i. Code feedback

1. Line-by-line explanation of the code
2. Generate natural language descriptions

ii. Code feedback Fix existing code

iii. Automatically predict code execution results based on unit tests

4.1 Summary of the inspection content

In summary, although the existing work involves some aspects, it does not meet our ability to test LLM's ability to generate project-level code. Therefore, based on this, we should mainly focus on how to accurately evaluate the following content in stages.

- Project planning (task breakdown) capability
 - Design document writing
 - Code base architecture design
 - Divide the project into several sub-modules and main operating files
 - What are the functions of each submodule?
 - What is the calling relationship between submodules?
 - UML Design Diagram
 - Determine what basic knowledge is needed (programming languages, algorithms, data structures, library functions, etc.)
- Generate codebase files file by file
- Ability to automatically generate unit tests
- Code review
 - The ability to predict code execution results (involves examining LLM's understanding of variables, internal memory to maintain variable operations, and logical reasoning ability, which is more difficult for non-GPT-4 models).

- Considering that automatically running unit tests may encounter difficulties, it is possible to consider using LLM to directly predict the current code execution result as a "**virtual unit test**" (Self-collaboration Code Generation via ChatGPT is also stated in this way, so they simply modify the code based on generated feedback)
 - Ability to automatically generate code feedback
 - Ability to modify code based on existing code and feedback
- Code execution

4.2 Evaluation details

4.2.1 project planning capacity

1. Code base architecture design/UML diagram design