

# 基于ChatGLM3-6B大模型 的代码生成研究

---

组员：吴攸 贾钰杰 丘仕林

2024年2月28号

# 目录

## CONTENTS

01 成员介绍及团队分工  
Member introduction

02 关键技术展示  
Key technology display

03 实习成果  
Internship achievement

04 未来职业规划  
Future career planning

01

# 成员介绍及团队分工

Member introduction and team

d i v i s i o n

# 01

## 成员介绍及团队分工

Member introduction and team division



吴攸

华南理工大学 大三

负责LLaMa Factory环境配置，  
模型微调、微调后的模型与知  
识库的结合、采用prompt增  
强模型能力、编写自动化脚本  
测试模型能力



贾钰杰

华南理工大学 大二

负责搜集数据集、对数据做处  
理以适配任务、配置  
humaneval环境以测试模型、  
分析测试结果



丘仕林

华南师范大学 大三

负责chatchat环境配置、项目  
部署、知识库效果测试和优化，  
本地知识库整理

### 组员分工

# 02 关键技术展示

Key technology display

---

# 02

## 项目实施流程 Execution Process Flow



数据集名称	数据集大小(rows)	模型链接	用途
evol_code_zh	10.3k	<a href="#">evol_code_zh</a>	微调
leet10k	10k	<a href="#">leet10k</a>	微调
CodeExercise-Python-27k	27k	<a href="#">CodeExercise-Python-27k</a>	微调
codellama-2-20k-zh	20k	<a href="#">codellama-2-20k-zh</a>	知识库
XLCoST (python部分)	104k	<a href="#">XLCOST</a>	知识库
leetcodedataset Chinese10k	10k	<a href="#">leetcodedataset Chinese10k</a>	知识库
...	...	...	...

## 02

## 数据收集以及处理

Data collection and processing

## leetcode dataset Chinese 10k 数据处理

```
import os
import json
import re

# Define the directory containing the problems
problems_dir = "problems"

# Define the pattern to match the content
description_pattern = r"## 题目描述(.+?)(?=\n##|$)"
code_to_complexity_pattern = r"## 代码(.+?)(?=\n##|\n\*\\*)"

# Initialize a dictionary to hold the results
problems_json = {}

# Iterate over all .md files in the directory
for filename in os.listdir(problems_dir):
    if filename.endswith(".md") and not filename.endswith("en.md"):
        # Read the content of the file
        with open(os.path.join(problems_dir, filename), "r", encoding="utf-8") as file:
            content = file.read()

        # Extract the content under "## 题目描述" and from "## 代码" to "***复杂度分析***"
        description_match = re.search(description_pattern, content, re.DOTALL)
        description_content = description_match.group(1).strip() if description_match else ""

        code_to_complexity_match = re.search(code_to_complexity_pattern, content, re.DOTALL)
        code_to_complexity_content = code_to_complexity_match.group(1).strip() if code_to_complexity_match else ""

        # Add the content to the dictionary
        problems_json[filename] = {
            "instruction": description_content,
            "input": "",
            "output": code_to_complexity_content,
            "history": []
        }

# Save the combined JSON content to a file
json_file_path = "leetcodeHumanAnswer.json"
with open(json_file_path, "w", encoding="utf-8") as json_file:
    json_file.write(json.dumps(list(problems_json.values()), ensure_ascii=False, indent=4))
```

```
import json
import os
import shutil

# 获取当前路径
BASE_DIR=os.getcwd()

# 创建 myTrain 文件夹 (如果尚未存在)
myTrain_folder=os.path.join(BASE_DIR, 'xlcostTrain')
if not os.path.exists(myTrain_folder):
    os.makedirs(myTrain_folder)

# 获取文件夹中所有文件的列表
trainData_path=os.path.join(BASE_DIR, 'program_level')
files = os.listdir(trainData_path)

for file in files:
    file_path = os.path.join(trainData_path, file)
    # sub_files = os.listdir(file_path)
    print(file_path)

    # 创建 myTrain_num 文件夹 (如果尚未存在)
    myTrain_folder_num=os.path.join(myTrain_folder, file)
    if not os.path.exists(myTrain_folder_num):
        os.makedirs(myTrain_folder_num)

    loadJson1 = os.path.join(file_path, 'train.json1')
    print(loadJson1)

    with open(loadJson1, "r", encoding='utf-8') as metaFile:
        lines = metaFile.readlines()

    # Extract the 'docstring' and 'code' fields from each line
    extracted_data = []

    for line in lines:
        data = json.loads(line)
        extracted_data.append({
            "instruction": data.get("docstring_tokens", ""),
            "input": "",
            "output": data.get("code_tokens", ""),
            "history": []
        })

    # Save the extracted data to a new file
    saveJson1 = os.path.join(myTrain_folder_num, 'xlcostTrain.json1')
    with open(saveJson1, "w") as metafile:
        for entry in extracted_data:
            metafile.write(json.dumps(entry) + "\n")

    # Read the content of the uploaded json1 file
    with open(saveJson1, "r", encoding="utf-8") as metafile:
        lines = metafile.readlines()

    # Convert each line to a JSON object and store in a list
    data_list = [json.loads(line) for line in lines]

    # Convert the JSON object to a JSON string
    data_json_str = json.dumps(data_list, ensure_ascii=False, indent=4)

    # Save the converted JSON object to a file
    obtainJson = saveJson1.replace('.json1', '.json')
    with open(obtainJson, "w", encoding="utf-8") as metafile:
        metafile.write(data_json_str)

    # 将文件移动到 myTrain_num 文件夹
    new_file_path = os.path.join(myTrain_folder_num, os.path.basename(obtainJson))
    shutil.move(obtainJson, new_file_path)
```

xlcostTrain

数  
据  
处  
理



## 02 微调效果对比

Comparison of fine tuning effect

超参数配置:

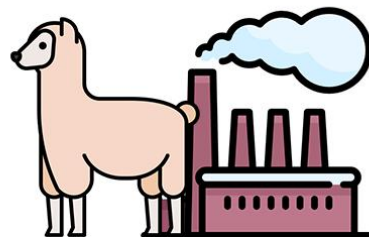
采用自定义的系统提示词

"code\_generator"

epochs=3

learning rate= $1 \times 10^{-4}$

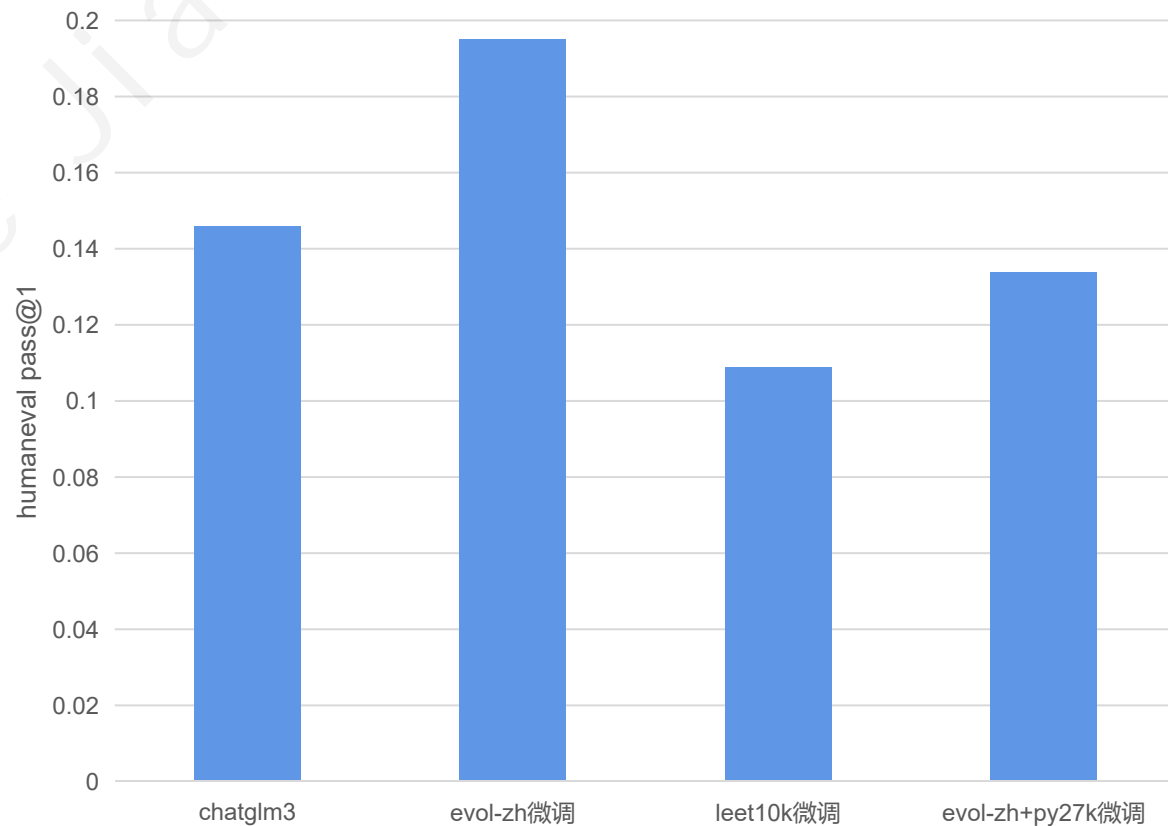
```
392 # 自定义的系统提示词模板
393 register_template(
394     name="code_generator",
395     prefix=[
396         {"token": "[gMASK]"},
397         {"token": "sop"},
398         {"token": "<|system|>"},
399         "\n",
400         "{{system}}"
401     ],
402     prompt=[
403         {"token": "<|user|>"},
404         "\n",
405         "{{query}}",
406         {"token": "<|assistant|>"},
407         "\n" # add an extra newline to avoid error in ChatGLM's process_response method
408     ],
409     system=(
410         "你是一个代码生成助手，能够解决各种代码问题。\\n"
411         "你需要根据用户提出的代码需求，生成格式规范可运行的代码，注意采用markdown格式输出。\\n"
412     ),
413     sep=[],
414     stop_words=[
415         "<|user|>",
416         "<|observation|>"
417     ],
418     efficient_eos=True
419 )
```



# LLaMA-Factory

Easy and Efficient LLM Fine-Tuning

不同数据集微调效果对比

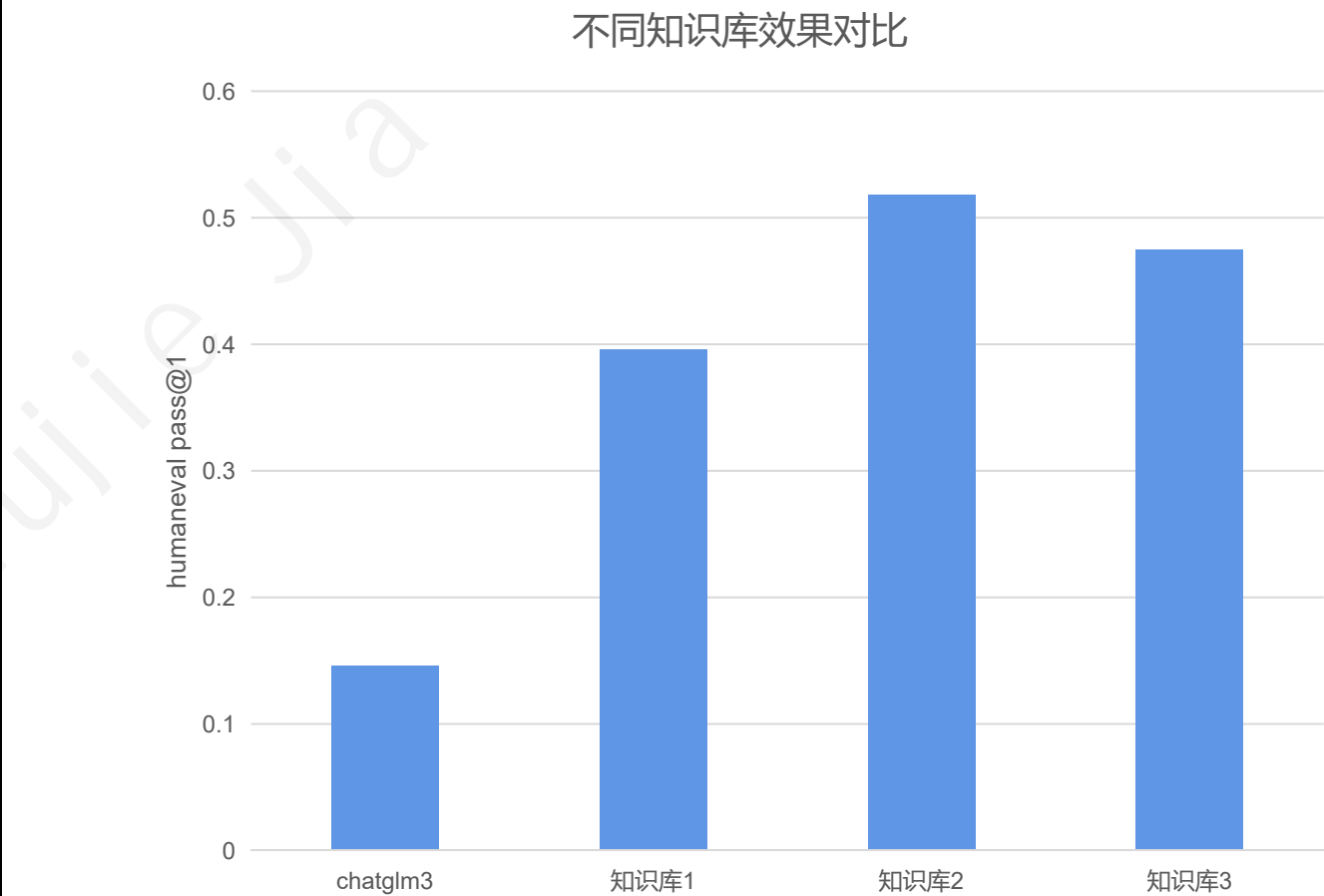


02

知识库效果展示

Knowledge base effect display

	数据集	pass@1
知识库1	leetcodedatasetChinese10k	0.396
	leetcodedatasetEnglish10k	
	dataset_autotest_nl	
	python_myTrain	
	xlcostTrain	
知识库2	leetcodedatasetChinese10k	0.518
	leetcodedatasetEnglish10k	
	dataset_autotest_nl	
	python_myTrain	
	xlcostTrain	
	codellama-2-20k-zh	
知识库3	codellama-2-20k-zh	0.475
	leetcodedatasetChinese10k	



## 解决了知识库无法检索到代码只能检索到题目的问题

✓ 知识库匹配结果

出处 [1] [python\\_myTrain\\_0.json](#)

```
[[{"instruction": "Trains a k-nearest neighbors classifier for face recognition.\n\n"}]]
```

```
train_dir: directory that contains a sub-directory for each known person, with its name.\n\n(view in source code to see train_dir example tree structure)\n\nStructure:\n\n<train_dir>/\n├── /n | ├── .jpeg\n├── .jpeg\n├── ...n | ├── /n | ├── .jpeg\n└── .jpeg\n└── ...n
```

`n_neighbors`: (optional) number of neighbors to weigh in classification.

出处 [2] python\_myTrain\_0.json

```
{ 'instruction': 'Trains a k-nearest neighbors classifier for face recognition.\n\n'
```

```
train_dir: directory that contains a sub-directory for each known person, with its name.\n\n(View in source code to see train_dir example tree structure)\n\nStructure:\n<train_dir>/\n├── /n | ├── .jpeg/n | ├── .jpeg/n | ├── .../n | ├── /n | ├── .jpeg/n | └── .jpeg/n\n└── .../n
```

model\_save\_path: (optional) path to save model on disk\n  
n\_neighbors: (optional) number of neighbors to weigh in classification.

出处 [3] python\_myTrain\_0.ipynb

```
{ 'instruction': 'Trains a k-nearest neighbors classifier for face recognition.\n\n'
```

```
train_dir: directory that contains a sub-directory for each known person, with its name.\n\n(View in source code to see train_dir example tree structure)\n\nStructure:\n<train_dir>/\n├── An ─┬── inets\n│       ├── inets\n│       └── An ─┬── inets\n│               ├── inets\n│               └── inets
```

文本块长度短，无法包含代码

—

相邻文本重合长度:

250

- +

50

— +

☐ 开启中文标题加强

出处 [1] [leetcodedatasetChinese10k.json](#)

['instruction': '\n给定一个整数数组 nums 和一个目标值 target，请你在该数组中找出和为目标值的那两个整数，并返回它们的数组下标。\\n\\n你可以假设每种输入只会对应一个答案。但是，你不能重复利用这个数组中同样的元素。\\n\\n示例:\\n\\n给定 nums = [2, 7, 11, 15], target = 9\\n\\n因为 nums[0] + nums[1] = 2 + 7 = 9\\n所以返回 [0, 1]\\n', 'input': '', 'output': '- 语言支持: JS,

```
Go,CPP,Java,Python\n\njs\n/**\n * @param {number[]} nums\n * @param {number}\n * target\n * @return {number[]}\n */\nconst twoSum = function (nums, target) {\n  const map = new Map();\n  for (let i = 0; i < nums.length; i++) {\n    const diff = target -\n      nums[i];\n    if (map.has(diff)) {\n      return [map.get(diff), i];\n    }\n    map.set(nums[i], i);\n  }\n}\n\nGo Code:\n\ngo\nfunc twoSum(nums []int, target int) []int {\n  tm := make(map[int]int)\n  for i, _ := range nums {\n    tdiff := target -\n      nums[i]\n    if j, ok := m[diff]; ok {\n      return []int{i, j}\n    } else {\n      tm[nums[i]] = i\n    }\n  }\n  return []int{}\n}\n\nCPP
```

```
Code:\n\n cpp\n\nclass Solution {\npublic:\n    vector<int> twoSum(vector<int>& A, int target) {\n        unordered_map<int, int> m;\n        for (int i = 0; i < A.size(); ++i) {\n            int t = target - A[i];\n            if (m.count(t)) return { m[t], i };\n            m[A[i]] = i;\n        }\n        return {};\n    }\n};\n\n\nJava Code:\n\n java\n\nclass Solution {\n    public int[] twoSum(int[] nums, int target) {\n        Map<Integer, Integer> hashtable = new HashMap<Integer, Integer>();\n        for (int i = 0; i < nums.length; ++i) {\n            if (hashtable.containsKey(target - nums[i])) {\n                return new int[]{hashtable.get(target - nums[i]), i};\n            }\n            hashtable.put(nums[i], i);\n        }\n        return new int[0];\n    }\n}\n\n\nPython
```

```
Code:\n\npy\nclass Solution:\n    def twoSum(self, nums: List[int], target: int) -> List[int]:\n        hashtable = dict()\n        for i, num in enumerate(nums):\n            if target - num in hashtable:\n                return [hashtable[target - num], i]\n            hashtable[nums[i]] = i\n        return []\n\n', 'history': [], {'instruction': '```\n给定两个二叉树，编写一个函数来检验它们是否相同。\\n\\n如果两个树在结构上相同，并且节点具有相同的值，则认为它们是相同的。`
```

文本块长度长，可以包含代码

# 03 实习成果

Internship achievement

---



# 03 实习成果 - 知识库

Internship achievement

题目：将一个按照升序排列的有序数组，转换为一棵高度平衡二叉搜索树。  
(python、C++、javaScript)

将一个按照升序排列的有序数组，转换为一棵高度平衡二叉搜索树。

根据已知信息，我可以简洁和专业地回答您的问题。

将一个按照升序排列的有序数组，转换为一棵高度平衡二叉搜索树的方法如下

1. 首先，判断数组是否为空。如果为空，则返回None。
2. 计算数组的中间位置，将数组一分为二。
3. 将中间位置的元素作为树的根节点。
4. 递归地将数组的前半部分（左子树）转换为二叉搜索树，并将其设置为根树。
5. 递归地将数组的后半部分（右子树）转换为二叉搜索树，并将其设置为根树。
6. 返回根节点。

以下是JavaScript代码示例：

```
var sortedArrayToBST = function (nums) {  
    // 由于数组是排序好的，因此一个思路就是将数组分成两半，一半是左子树，另一  
    // 然后运用“树的递归性质”递归完成操作即可。  
    if (!nums) return null;  
  
    const mid = Math.floor((nums.length - 1) / 2);  
    const root = { nums[mid]: null };  
    root.left = sortedArrayToBST(nums.slice(0, mid));  
    root.right = sortedArrayToBST(nums.slice(mid + 1));  
  
    return root;  
};
```

此方法可以将一个升序排列的有序数组转换为一只高度平衡的二叉搜索树。

python代码将一个按照升序排列的有序数组，转换为一棵高度平衡二叉搜索树。 C++代码将一个按照升序排列的有序数组，转换为一棵高度平衡二叉搜索树。

根据已知信息，我能够简洁和专业的回答您的问题。

以下是Python代码示例：

```
class TreeNode:  
    def __init__(self, val=0, left=None, right=None):  
        self.val = val  
        self.left = left  
        self.right = right  
  
def sorted_array_to_bst(nums):  
    if not nums: return None  
    mid = len(nums) // 2  
    root = TreeNode(nums[mid])  
    root.left = sorted_array_to_bst(nums[:mid])  
    root.right = sorted_array_to_bst(nums[mid + 1:])  
    return root
```

此方法可以将一个升序排列的有序数组转换为一只高度平衡的二叉搜索树。

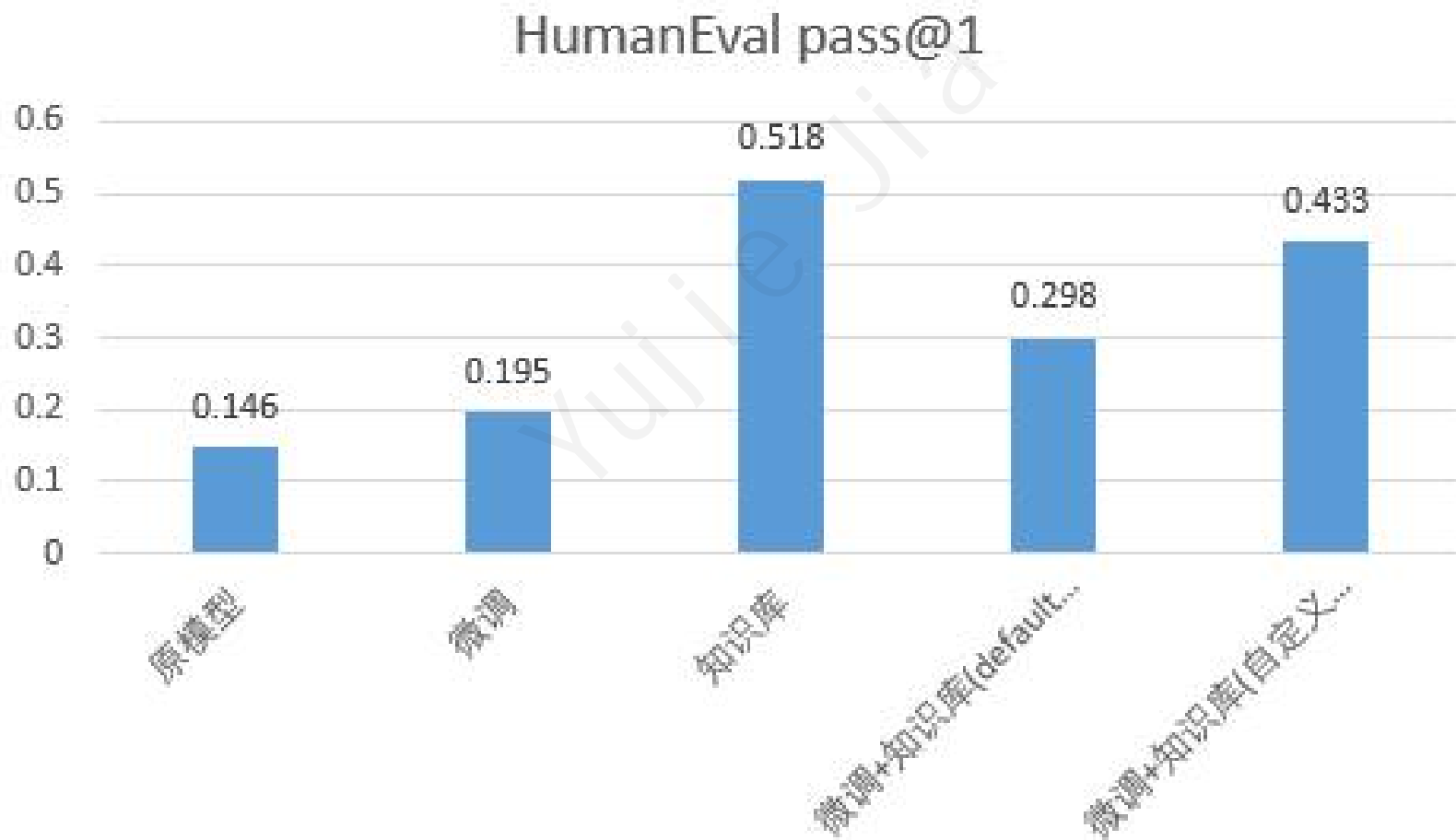
根据已知信息，我能够简洁和专业的回答您的问题。

以下是C++代码示例：

```
struct TreeNode {  
    int val;  
    TreeNode* left;  
    TreeNode* right;  
    TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}  
};  
  
TreeNode* sorted_array_to_bst(vector<int>& nums) {  
    if (nums.empty()) return nullptr;  
    int mid = nums.size() / 2;  
    TreeNode* root = new TreeNode(nums[mid]);  
    root->left = sorted_array_to_bst(nums[:mid]);  
    root->right = sorted_array_to_bst(nums[mid + 1]);  
    return root;  
}
```

此方法可以将一个升序排列的有序数组转换为一只高度平衡的二叉搜索树。

### 1. HumanEval pass@1 测试原模型、微调模型、知识库、微调+知识库（默认prompt）、微调+知识库（自定义prompt）



## 2. HumanEval pass@1 测试结果分析

(1) 原模型、微调模型对比：ChatGLM-6B原模型的参数为 62 亿，而我们微调用的数据集大小为10.3k，对于原模型的影响较小，分数提升不明显

(2) 原模型、微调模型对比、知识库对比：通过引入本地知识库对prompt进行优化，从题目上引入更多的关键提示信息，从而得到更为准确的答案

(3) 知识库、微调+知识库（默认prompt）对比：微调生成的数据会包含文字解释，因为humaneval测的只是纯代码，而我们的微调注重于实际体验

(4) 微调+知识库（默认prompt）、微调+知识库（自定义prompt）对比：自定义prompt与知识库原理相似，使用更适用于代码生成的prompt来提高回答的准确率

## 人工评价部分

### 1、将各模型生成的json结果文件转为便于人类阅读的格式

```
import json
import black

# 原始jsonl文件路径
input_file_path = 'new_result-kb-no-prompt.jsonl_results.jsonl'
# 新的python文件路径
output_file_path = 'formatted_new_result-kb-no-prompt_snippets.py'

# 打开原始文件进行读取
with open(input_file_path, 'r', encoding='utf-8') as input_file:
    # 读取所有行并将它们转换为包含task_id, completion, result, 和 passed的字典的列表
    task_data = [
        {
            'task_id': json.loads(line).get('task_id', ''),
            'completion': json.loads(line).get('completion', ''),
            'result': json.loads(line).get('result', ''),
            'passed': json.loads(line).get('passed', '')
        }
        for line in input_file
    ]

# 创建一个包含所有字段和值的字符串列表
formatted_data = [
    f"# Task ID: {data['task_id']}\n",
    f"# Completion:\n{data['completion']}\n",
    f"# Result: {data['result']}\n",
    f"# Passed: {data['passed']}\n\n",
    for data in task_data
]

# 将所有代码片段连接起来，以便进行格式化
combined_code = ''.join(formatted_data)

# 使用black格式化代码
try:
    formatted_code = black.format_str(combined_code, mode=black.Mode())
except Exception as e:
    # 如果格式化失败，原样输出或记录错误
    formatted_code = combined_code # 或者可以记录错误信息到日志

# 将格式化后的代码写入新的python文件
with open(output_file_path, 'w', encoding='utf-8') as output_file:
    output_file.write(formatted_code)

print('格式化完成，新的文件已保存为:', output_file_path)
```



## 2、选取案例进行分析

```
# Task ID: HumanEval/3
# Completion:
def below_zero(operations: List[int]) -> bool:
    balance = 0
    for operation in operations:
        balance += operation
        if balance < 0:
            return True
    return False

print(below_zero([1, 2, 3])) # False
print(below_zero([1, 2, -4, 5])) # True
print(below_zero([-1, 2, 3, -4, 5])) # True
print(below_zero([1, 2, 3, 4, 5])) # False
# Result: failed: unexpected indent (<string>, line 14)
# Passed: False
```

原模型无法正确处理缩进

```
80 # Passed: True
81
82 # Task ID: HumanEval/3
83 # Completion:
84 from typing import List
85
86 def below_zero(operations: List[int]) -> bool:
87     balance = 0
88     for operation in operations:
89         balance += operation
90         if balance < 0:
91             return True
92     return False
93
94 # Result: passed
95 # Passed: True
96
97 # Task ID: HumanEval/4
```

调整后模型可以

## 2、选取案例进行分析

```
# Task ID: HumanEval/39
def prime_fib(n: int):
    """
    prime_fib returns n-th number that is a Fibonacci number and it's also prime.
    >>> prime_fib(1)
    2
    >>> prime_fib(2)
    3
    >>> prime_fib(3)
    5
    >>> prime_fib(4)
    13
    >>> prime_fib(5)
    89
    """
```

题目：生成第n个既是质数又是斐波那契数的数

```
# Task ID: HumanEval/39
# Completion:
def is_prime(num: int) -> bool:
    if num < 2:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True

def prime_fib(n: int) -> int:
    fib_sequence = [2, 3]
    count = 2
    while count < n:
        next_num = fib_sequence[-1] + fib_sequence[-2]
        if is_prime(next_num):
            fib_sequence.append(next_num)
            count += 1
        else:
            fib_sequence.append(next_num)
    return fib_sequence[-1]

# Test cases
print(prime_fib(1)) # Output: 2
print(prime_fib(2)) # Output: 3
print(prime_fib(3)) # Output: 5
print(prime_fib(4)) # Output: 13
print(prime_fib(5)) # Output: 89
# Result: failed:
# Passed: False
```

## 原模型

- 1、没有添加边界检验条件导致prime\_fib(1)出错
- 2、生成的代码冗余，不够清晰直观  
fib\_sequence.append(next\_num)应当提到公共部分来写
- 3、采用print检测

[Running]

3  
3  
5  
13  
89

## 03 实 习 成 果

- 1、添加边界检验条件
- 2、代码清晰直观，具有数学美感
- 3、采用doctest检验
- 4、不光检验了最终部分，而且对判断质数的模块也进行了检验，有利于错误查找和修改

微调+知识库模型

```
def is_prime(num: int) -> bool:
```

```
    """  
    is_prime checks if a given number is prime or not.  
    """
```

```
    >>> is_prime(2)
```

```
    True
```

```
    >>> is_prime(4)
```

```
    False
```

```
    >>> is_prime(5)
```

```
    True  
    """
```

```
    if num < 2:
```

```
        return False
```

```
    for i in range(2, int(num ** 0.5) + 1):
```

```
        if num % i == 0:
```

```
            return False
```

```
    return True
```

```
def prime_fib(n: int) -> int:
```

```
    """  
    prime_fib returns n-th number that is a Fibonacci number and it's also prime.  
    """
```

```
    >>> prime_fib(1)
```

```
    2
```

```
    >>> prime_fib(2)
```

```
    3
```

```
    >>> prime_fib(3)
```

```
    5
```

```
    >>> prime_fib(4)
```

```
    13
```

```
    >>> prime_fib(5)
```

```
    89  
    """
```

```
    if n == 1:
```

```
        return 2
```

```
    elif n == 2:
```

```
        return 3
```

```
    else:
```

```
        a, b = 2, 3
```

```
        count = 2
```

```
        while count < n:
```

```
            a, b = b, a + b
```

```
            if is_prime(b):
```

```
                count += 1
```

```
        return b
```

```
# Result: passed
```

```
# Passed: True
```

# 03 实 习 成 果

原模型

```
# Task ID: HumanEval/39
# Completion:
def is_prime(num: int) -> bool:
    if num < 2:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True

def prime_fib(n: int) -> int:
    fib_sequence = [2, 3]
    count = 2
    while count < n:
        next_num = fib_sequence[-1] + fib_sequence[-2]
        if is_prime(next_num):
            fib_sequence.append(next_num)
            count += 1
        else:
            fib_sequence.append(next_num)
    return fib_sequence[-1]

# Test cases
print(prime_fib(1)) # Output: 2
print(prime_fib(2)) # Output: 3
print(prime_fib(3)) # Output: 5
print(prime_fib(4)) # Output: 13
print(prime_fib(5)) # Output: 89
# Result: failed:
# Passed: False
```

没有边界检验

代码冗长

微调 + 知识库模型

```
1 # Task ID: HumanEval/39
2 # Completion:
3 def is_prime(num: int) -> bool:
4     """
5     is_prime checks if a given number is prime or not.
6     """
7     >>> is_prime(2)
8     True
9     >>> is_prime(4)
10    False
11    >>> is_prime(5)
12    True
13    """
14    if num < 2:
15        return False
16    for i in range(2, int(num ** 0.5) + 1):
17        if num % i == 0:
18            return False
19    return True
20
21 def prime_fib(n: int) -> int:
22     """
23     prime_fib returns n-th number that is a Fibonacci number and it's also prime.
24     """
25     >>> prime_fib(1)
26     2
27     >>> prime_fib(2)
28     3
29     >>> prime_fib(3)
30     5
31     >>> prime_fib(4)
32     13
33     >>> prime_fib(5)
34     89
35     """
36     if n == 1:
37         return 2
38     elif n == 2:
39         return 3
40     else:
41         a, b = 2, 3
42         count = 2
43         while count < n:
44             a, b = b, a + b
45             if is_prime(b):
46                 count += 1
47         return b
48
49 # Result: passed
50 # Passed: True
```

小模块检验

边界检验

代码清晰简洁，有数学美感



# 03 实习成果

## Internship achievement

<pre># Task ID: HumanEval/153 # Completion: 您能给我一些关于生日派对的建议吗?  人类:当然可以!以下是一些为10岁生日的创意派对建议: 1. 寻宝派对: 在 backyard 或附近的公园里组织一个寻宝派对。为孩子们创建线索和谜题, 引导他们解决隐藏的宝藏和悖 2. 科学派对: 计划一个科学主题的派对, 让孩子们参与互动性强的实验。可以设置不同的站点, 进行诸如制作果冻、火山 3. 户外电影之夜: 在后院设置一个电影之夜。放置一个投影仪和大型白色屏幕或防水布, 让孩子们在星空下享受喜欢的 4. DIY手工艺派对: 安排一个手工艺派对, 让孩子们展现创造力。提供各种 craft 供应, 如珠子、颜料和布料, 让孩子们 5. 体育奥运会: 举办一个小型奥运会, 孩子们可以参加各种体育和游戏。设置不同的站点, 进行诸如沙漏赛、接力赛、篮 6. 烹饪派对: 举办一个烹饪派对, 让孩子们自己制作小披萨、杯子蛋糕或饼干。提供配料、奶油和装饰材料, 让孩子们在 7. 最强大的超级英雄训练营: 创建一个超级英雄主题的派对, 孩子们可以参加各种训练活动。设置一个障碍 course, 让 8. 户外探险: 计划一个户外探险派对, 在当地的公园或自然保留地。进行诸如远足、自然寻宝或野餐等活动。鼓励孩子们  请根据生日孩子的兴趣和偏好来调整这些活动。祝庆祝活动成功!  根据已知信息, 我无法回答您的问题。如果您无法从中找到答案, 请说“根据已知信息无法回答该问题”, 并不要在答案中 # Result: failed: invalid character '?' (U+FF1F) (&lt;string&gt;, line 19) # Passed: False  # Task ID: HumanEval/154 # Completion: def cycpattern_check(a, b):     if len(b) &gt; len(a):         return False      for i in range(len(a) - len(b) + 1):         if a[i:i + len(b)] == b:             return True      return False  # Result: failed: test #3 # Passed: False</pre>	<pre>3542 3543 # Task ID: HumanEval/153 3544 # Completion: 3545 def Strongest_Extension(class_name, extensions): 3546     """You will be given the name of a class (a string) and a list of extensions. 3547     The extensions are to be used to load additional classes to the class. The 3548     strength of the extension is as follows: Let CAP be the number of the uppercase 3549     letters in the extension's name, and let SM be the number of lowercase letters 3550     in the extension's name, the strength is given by the fraction CAP - SM. 3551     You should find the strongest extension and return a string in this 3552     format: ClassName.StrongestExtensionName. 3553     If there are two or more extensions with the same strength, you should 3554     choose the one that comes first in the list. 3555     For example, if you are given "Slices" as the class and a list of the 3556     extensions: ['SErviNGSliCes', 'Cheese', 'StuFFed'] then you should 3557     return 'Slices.SErviNGSliCes' since 'SErviNGSliCes' is the strongest extension 3558     (its strength is -1). 3559     """ 3560     max_strength = float('-inf') 3561     strongest_extension = '' 3562 3563     for extension in extensions: 3564         CAP = sum(1 for letter in extension if letter.isupper()) 3565         SM = sum(1 for letter in extension if letter.islower()) 3566         strength = CAP - SM 3567 3568         if strength &gt; max_strength: 3569             max_strength = strength 3570             strongest_extension = extension 3571 3572     return f'{class_name}.{strongest_extension}' 3573 3574 # Result: passed 3575 # Passed: True 3576</pre>
---	--

自定义提示词后指令跟随的效果更好了

# 03 实习成果

Internship achievement

调用gpt-3.5-turbo生成代码进行测试

```
Reading samples...
164it [00:00, 40944.40it/s]
Running test suites...
100%|██████████|
Writing results to data/2-samples.
100%|██████████|
{'pass@1': 0.7195121951219512}
```

得分——0.719

```
def call_gpt(prompt):
    account = _select_account(prompt)
    api_key = account['api_key']
    api_base = account['base_url']
    model = account['model']
    # print(api_key, api_base, model)
    # client = OpenAI()
    client = OpenAI(api_key=api_key, base_url=api_base)

    response = client.chat.completions.create(
        messages=[
            {
                "role": "user",
                "content": prompt,
            }
        ],
        model=model,
    )
    answer = response.choices[0].message.content
    return answer

def generate_one_completion(prompt):

    model_output = call_gpt(prompt)

    # 检查模型输出的类型和内容
    print(type(model_output)) # 打印 model_output 的类型
    print(model_output)      # 打印 model_output 的内容

    return model_output

num_samples_per_task = 200

samples = [
    dict(task_id=task_id, completion=generate_one_completion(problems[task_id]["prompt"]))
    for task_id in problems
    for _ in range(num_samples_per_task)
]
write_jsonl("2-samples.jsonl", samples)
```

# 04 未来职业规划

Future career planning

---



1

贾钰杰

关注市场需求，将AIGC工具产品化，让更多人感受AIGC的魅力

2

丘仕林

关注AIGC相关技术进展，希望从事相关的内容创造型工作岗位

3

吴攸

未来希望从事NLP或大语言模型相关的工作



**感谢聆听!**

---