

ClassEval: A Manually-Crafted Benchmark for Evaluating LLMs on Class-level Code Generation

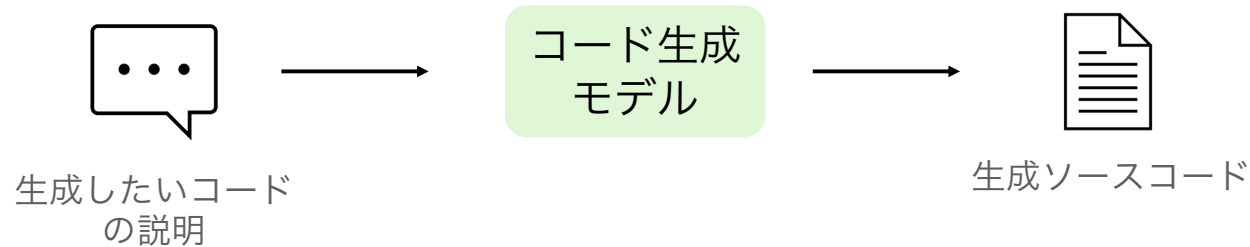
Xueying Du Mingwei Liu Kaixin Wang Hanlin Wang Junwei Liu
Yixuan Chen Jiayi Feng Chaofeng Sha Xin Peng Yiling Lou
Fudan University
Shanghai, China

ICSE '24: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering May 2024,
Article No.: 227, Pages 1–12

背景 | LLMによるコード生成

コード生成

与えられ得た自然言語の説明に対して、自動的にコードスニペットを生成



開発生産性の向上に貢献

LLMの進歩

LLMは、自然言語に限らず、コード生成の分野でも良い性能を示ように

→コード生成分野に大きな前進をもたらした

LLMのコード生成能力の評価

LLMによるコード生成が進歩する傍ら，LLMのコード生成能力を評価する取り組みが行われる

理由：

- ・ LLMの進歩の評価
- ・ LLMのコード生成能力の特性や限界の理解
- ・ 実用性の検討 など

一般的に，コード生成ベンチマークで評価される

コード生成ベンチマーク

- LLMのコード生成能力を評価するための問題集
- 実際にコードを生成し，どれくらいテストに合格するかでLLMを評価

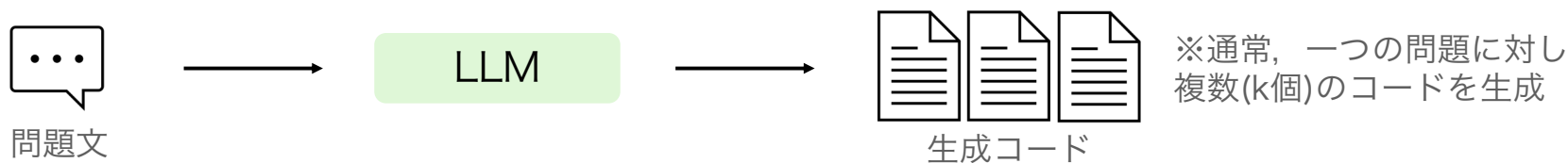
HumanEval | 既存の評価ベンチマーク

OpenAIが発表した、Pythonコード生成能力を評価するためのベンチマーク

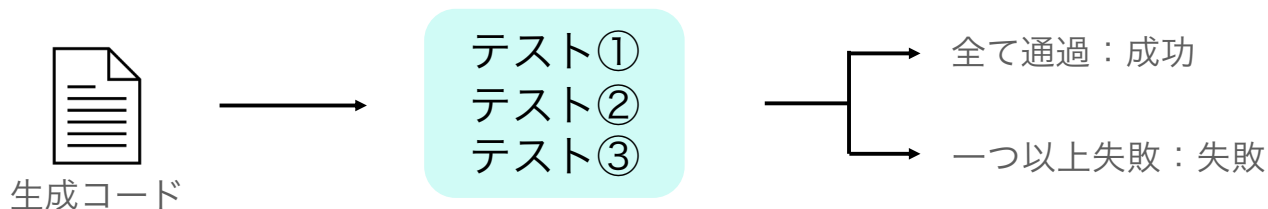
HumanEvalは、<問題文, テストケース>を含むタスク164個で構成されている

評価手順

①タスクの**問題文**をLLMに入力



②生成されたコードを**テストケース**で評価



③Pass@kなどの指標で正解率を算出

既存ベンチマークの問題点

| 単一の関数を生成するタスクで構成されている

そのため、LLMの以下のような能力が十分に評価できない

- ・ 長いコードの生成能力
 - 生成コードが比較的短いため(平均十数行程度)
- ・ 依存関係を持つコードの評価
 - 独立した関数の生成が対象なため

| 実際のソフトウェア開発では、より複雑で長いコードが記述される

→ 独立した関数の生成能力の評価では不十分

目的・アプローチ

目的

より複雑で難易度の高いコード生成をする際の、LLMのコード生成能力の評価

アプローチ

クラスレベルのコード生成ベンチマーク, “ClassEval”を手動で構築

→長いコードや依存関係を含む, クラスレベルでのコード生成能力を評価できる

ClassEval

LLMのクラスレベルでのコード生成能力を評価するための新たなベンチマーク

-評価手順はHumanEvalと基本的に同じ(コード生成→テストで評価)

100のPythonクラスレベルのタスクで構成

タスクは主に、<クラススケルトン, テスト>で構成

- ・クラススケルトン

- 構造化されたLLMへの入力形式

- HumanEvalでいう”問題文”に相当する

- 生成するコードの情報(クラス名, シグネチャ, 説明など)がまとめられている

- ・テスト

- 生成されたクラスの正確性を評価

- クラスレベルのテストとメソッドレベルのテスト両方を含む

クラススケルトン

生成すべきクラスの構造と要件が定義された入力形式

<pre>from datetime import datetime</pre>	Import Statements
<pre>class VendingMachine:</pre>	Class Name
<pre> """This is a class to simulate a vending machine, including adding products, inserting coins, purchasing products, viewing balance, replenishing product inventory, and displaying product information. """</pre>	Class Description
<pre> def __init__(self):</pre>	Class Constructor
<pre> """ Initializes the vending machine's inventory and balance. """</pre>	
<pre> self.inventory = {} self.balance = {}</pre>	
<pre> def purchase_item(self, item_name):</pre>	Method Signature
<pre> """ Purchases a product from the vending machine and returns the balance after the purchase.</pre>	Functional Description
<pre> :param item_name: str, the name of the product to be purchased, which should be in the vending machine. :return: If successful, returns the balance of the vending machine after the product is purchased, float, if the product is out of stock, returns False.</pre>	Parameter/Return
<pre> >>> vendingMachine.inventory = {'Coke': {'price': 1.25, 'quantity': 10}} >>> vendingMachine.balance = 1.25 >>> vendingMachine.purchase_item('Coke') 0.0 >>> vendingMachine.inventory {'Coke': {'price': 1.25, 'quantity': 9}} """</pre>	Description
<pre> def restock_item(self, item_name, quantity):</pre>	Method Signature
<pre> """</pre>	Functional Description
<pre> Replenishes the inventory of a product already in the vending machine. :param item_name: The name of the product to be replenished, str, which should be in the vending machine. :param quantity: The quantity of the product to be replenished, int, which is greater than 0. :return: If the product is already in the vending machine, returns True, otherwise, returns False.</pre>	Parameter/Return Description
<pre> >>> vendingMachine.inventory = {'Coke': {'price': 1.25, 'quantity': 10}} >>> vendingMachine.restock_item('Coke', 10) True >>> vendingMachine.inventory {'Coke': {'price': 1.25, 'quantity': 20}} """</pre>	Example Input/Output
<pre>...</pre>	

クラスレベルの情報

- ・クラス名
- ・クラスの説明(自然言語)
- ・import文
- ・コンストラクタ

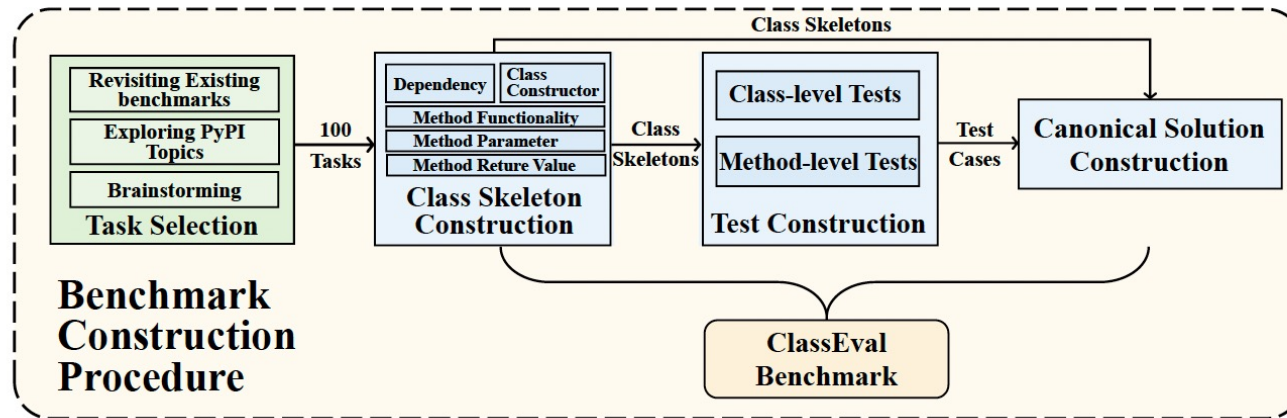
メソッドレベルの情報

- ・シグネチャ
- ・メソッドの説明(自然言語)
- ・パラメータ/戻り値の説明
- ・入出力の例

を含む

これを入力として、コード生成を行う

ClassEval構築手順



①タスクの選定

- どんなタスクを用意するかを決定
- 既存ベンチマークや既存リポジトリの利用, 話し合いなど

②クラススケルトンの作成

- 選定したタスクを元に,
前述の情報を規定の形式で記述

③テストケースの作成

- クラスレベル・メソッドレベルの
テストを作成

④正解例コードの作成

- テストケースの妥当性評価用?

全て手動で作成

ClassEvalの特徴

HumanEvalとの主な相違点

①コードの長さ

-HumanEval：平均行数11.5の比較的短いコード

-ClassEval：平均行数45.7行の比較的長いコード

②依存関係

-HumanEval：独立した関数が生成対象

-ClassEval：クラス内のメソッド間の依存関係を持つクラスが生成対象

生成タスクの内容

Topic	Description	Examples	#Tasks
Management Systems	Operational functionalities in common software management systems projects	Student Registration System, Movie Booking System	27
Data Formatting	Processing data according to specific rules or patterns	Text-to-number Conversion, URL Format Validation	26
Mathematical Operations	Algorithms for mathematical and statistical problems	Basic Arithmetic Operations, Area Calculation	16
Game Development	Algorithms for game functionalities, including mechanics and state management	Minesweeper Game, Gomoku Game	10
File Handling	Common file operations including reading, writing, and simple processing data in files	CSV File Processor, JSON File Processor	9
Database Operations	Implementation of common database operations	Library Database Operation, SQL Query Generator	7
Natural Language Processing	Techniques for processing and analyzing text data	Stop Word Removal, Longest Word Identification	5

管理システム， データ処理， ゲーム管理， 自然言語処理など幅広く

RQ1

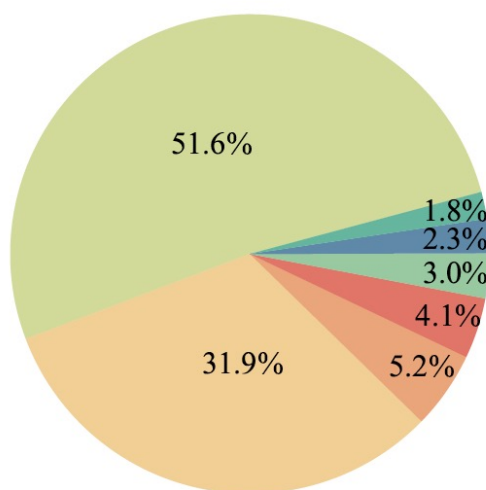
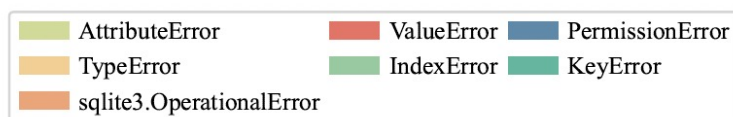
クラスレベルのコード生成において、既存のLLMはどのくらいの性能？

Model	Class-level			Method-level		
	Pass@1	Pass@3	Pass@5	Pass@1	Pass@3	Pass@5
GPT-4	37.6%	41.3%	42.0%	62.8%	67.4%	68.5%
GPT-3.5	29.6%	34.9%	36.0%	50.4%	59.0%	61.1%
WizardCoder	12.2%	20.0%	23.0%	35.2%	47.1%	51.1%
Instruct-StarCoder	10.2%	12.7%	14.0%	23.1%	26.5%	27.7%
SantaCoder	8.6%	9.9%	10.0%	27.7%	33.0%	34.9%
Instruct-CodeGen	8.2%	12.3%	13.0%	24.9%	34.3%	37.1%
CodeGeeX	7.2%	9.4%	10.0%	21.2%	27.1%	29.5%
InCoder	6.2%	7.6%	8.0%	21.1%	26.5%	29.1%
Vicuna	3.0%	3.6%	4.0%	11.0%	15.8%	18.4%
ChatGLM	1.4%	2.6%	3.0%	8.2%	11.2%	12.4%
PolyCoder	1.4%	2.2%	3.0%	13.2%	17.5%	19.6%

- ・全てのLLMにおいて、メソッドレベルよりクラスレベルのタスクの方が大幅に低い性能を示した
- ・GPT-4とGPT-3.5が他を大幅に上回る性能を示した

RQ4

生成に失敗したコードの主なエラー要因は？



AttributeError：オブジェクトが持つ属性に関するエラー

TypeError：型に関するエラー

が大半を占める

- ・ LLMは，属性や型を正しく認識できていないことが多い
- ・ 依存関係が関わる際に，属性や引数の型をうまく受け渡しできないケースが観察された

まとめ

- | LLMのクラスレベルのコード生成能力を評価するベンチマークClassEvalを作成

- | LLMは軒並み、クラス生成能力はメソッド生成能力を大きく下回る結果となった

所感

- | クラステンプレートの内容の妥当性
 - 内容は十分or不十分or過多？