

20241107

HumanEvalの問題の分類による評価への影響の調査

鳥飼優有

背景 プログラム合成

●仕様や条件からシステムがプログラムコードを自動生成

●プログラム合成のアプローチ

◆形式的手法

- ・ 論理的な記述からプログラムコードを生成
- ・ 結局複雑な記述は要求され，本末転倒な一面も

◆複数の入出力からプログラムを予測する手法

- ・ ExcellのFrashFillなど

◆自然言語を理解して生成する手法

- ・ chatGPTなど，大規模言語モデルにより飛躍的に進化

背景 LLMによるプログラム合成

- プログラム合成が可能なLLMが多数登場し、性能は日々進化している

- LLMのプログラム合成能力を評価するベンチマークもいくつか提案されている

 - ◆ MBPP (Mostly Basic Python Problems Dataset / Google Research)

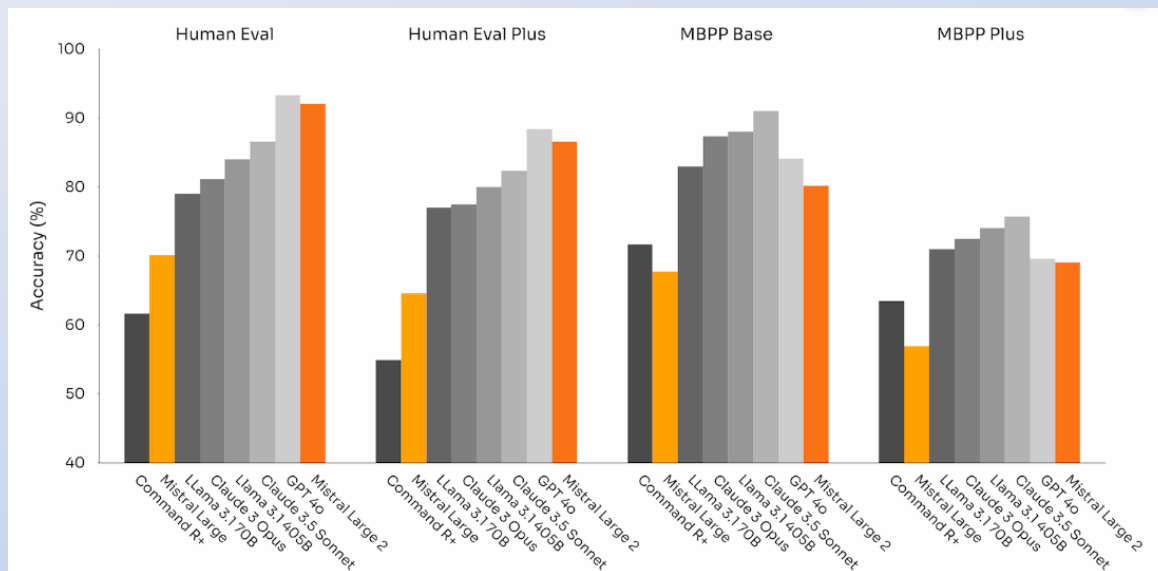
 - ・ 974個の基礎～中級レベルプログラミング問題

 - ◆ Human Eval (OpenAI)

 - ・ 164個の複雑なプログラミング問題

問題 ベンチマークの比較方法

- 現状LLMのプログラム合成能力はベンチマークによる総合評価によって比較される
- ベンチマークの問題の種類によって、LLMには得意不得意がある可能性がある
- どのようなタスクによってどのLLMが有用なのかを明らかにすることで、最適なLLMの選択を支援できる



出典：

<https://mistral.ai/news/mistral-large-2407/>

アプローチ

●分類項目を決める

- ・ まずHumanEval全体から，どのように分類すべきか分類項目を決める必要がある

●HumanEvalの164個の問題を分類する

- ・ HumanEvalのそれぞれの問題を最も当てはまる項目に分類する

●分類によって統計情報に母集団との有意差がないか調べる（仮）

- ・ t検定やカイ二乗検定など

HumanEval

ダウンロード：

https://huggingface.co/datasets/openai/openai_humaneval

OpenAIの164のプログラミング問題のセット

例： HumanEval/14

・ id

```
HumanEval/14
```

・ 問題文(プロンプト)

```
from typing import List
def all_prefixes(string: str) -> List[str]:
    """ Return list of all prefixes from
    shortest to longest of the input string
    >>> all_prefixes('abc')
    ['a', 'ab', 'abc']
    """
```

・ エントリーポイント

```
all_prefixes
```

・ 解の形式

```
result = []
for i in range(len(string)):
    result.append(string[:i+1])
return result
```

・ ユニットテスト

```
METADATA = {
    'author': 'jt',
    'dataset': 'test'
}
def check(candidate):
    assert candidate('') == []
    assert candidate('asdfgh') ==
    . . .
```

手書きで作成

- ・ LLMは公開されているコードで学習されているため，既存のコードを参照せずに作成された

pass@k

$$\text{pass}@k := \mathbb{E}_{\text{Problems}} \left[1 - \frac{\binom{n-c}{k}}{\binom{n}{k}} \right]$$

k個の中で1つでもユニットテストに合格していればその集合を合格と、
みなして、合格した集合の割合を算出する

pass@3の例



$$\text{Pass}@3 = 2/4 = 0.5$$

エントリーポイントによる分類

●全てのHumanEvalの問題のエントリーポイントをchatGPTに与えて分類させる

以下のリストの要素を内容で何らかの分類を行い、pythonで処理できるようなインデックスのリストの形で出力してください

```
['has_close_elements',  
'separate_paren_groups',  
'truncate_number',  
'below_zero',  
'mean_absolute_deviation',  
'intersperse',  
'parse_nested_parens',  
'filter_by_substring',  
'sum_product',  
'rolling_max',  
'make_palindrome',  
'string_xor',  
'longest',  
'greatest_common_divisor',  
'all_prefixes',  
'string_sequence',
```



リストの要素を以下のように分類しました：

- **数学的操作 (math_operations):** インデックス [8, 13, 24, 25, 31, 36, 39, 40, 43, 45, 49, 53, 59, 60, 62, 71, 75, 82, 85, 97, 122, 133, 138, 139, 142, 149]
- **文字列操作 (string_manipulation):** インデックス [7, 10, 11, 15, 16, 27, 38, 48, 50, 51, 54, 64, 93, 101, 107, 117, 118, 125, 134, 143, 162]
- **ソートアルゴリズム (sorting_algorithms):** インデックス [9, 19, 33, 35, 37, 70, 88, 114, 115, 116, 120, 126, 129, 135, 147, 156, 158]
- **論理操作 (logical_operations):** インデックス [20, 32, 56, 61, 69, 74, 76, 80, 91, 95, 96, 98, 108, 113, 119, 132, 137, 141, 152, 154, 155]
- **その他 (others):** インデックス [0, 1, 2, 3, 4, 5, 6, 12, 14, 17, 18, 21, 22, 23, 26, 28, 29, 30, 34, 41, 42, 44, 46, 47, 52, 55, 57, 58, 63, 65, 66, 67, 68, 72, 73, 77, 78, 79, 81, 83, 84, 86, 87, 89, 90, 92, 94, 99, 100, 102, 103, 104, 105, 106, 109, 110, 111, 112, 121, 123, 124, 127, 128, 130, 131, 136, 140, 144, 145, 146, 148, 150, 151, 153, 157, 159, 160, 161, 163]

エントリーポイントによる分類 結果

●分類項目

- ・ 数学的操作
- ・ 文字列操作
- ・ ソートアルゴリズム
- ・ 論理操作

●分類結果

分類項目	数学的操作	文字列操作	ソートアルゴリズム	論理操作	その他
個数	26	21	17	21	79

プロンプトによる分類

● エントリーポイントの分類項目を使用

- ・ HumanEvalの全ての問題のプロンプトをchatGPTに入力するとトークン数が多くなり、分類性能が落ちると考えられる
- ・ エントリーポイントの分類項目をそのまま使用する

● プロンプトにより1つずつ分類する

- ・ 問題ごとにエントリーポイントよりも詳細な内容であるプロンプトをchatGPTに参照させ、分類を行う



この問題は「数学的操作」に分類するのが適当です。理由は、関数が整数の配列から各整数の桁の和を計算し、その結果に基づいて条件を評価するためです。数値の性質を操作して、その合計を調べるという数学的な処理が含まれているため、数学的操作のカテゴリに該当します。

プロンプトによる分類 結果

●分類結果

分類項目	数学的操作	文字列操作	ソートアルゴリズム	論理操作
個数	81	47	10	26

数学的操作	文字列操作	ソートアルゴリズム	論理操作
0, 2, 4, 8, 9, 13, 20, 21, 24, 25, 31, 32, 35, 36, 39, 41, 42, 44, 45, 46, 47, 49, 50, 53, 55, 58, 59, 60, 62, 63, 66, 67, 68, 69, 71, 73, 75, 76, 77, 79, 82, 83, 84, 90, 92, 94, 96, 97, 99, 100, 102, 103, 106, 107, 108, 114, 115, 121, 122, 123, 127, 128, 130, 131, 133, 136, 138, 139, 142, 144, 147, 148, 150, 151, 152, 155, 157, 159, 160, 162, 163	1, 5, 6, 7, 10, 12, 14, 15, 16, 17, 18, 23, 27, 28, 29, 38, 48, 51, 64, 65, 74, 78, 80, 86, 89, 91, 93, 95, 98, 101, 111, 112, 113, 117, 118, 124, 125, 134, 140, 141, 143, 149, 153, 154, 156, 158, 161	19, 33, 34, 37, 70, 88, 105, 116, 120, 145	3, 11, 22, 26, 30, 40, 43, 52, 54, 56, 57, 61, 72, 81, 85, 87, 104, 109, 110, 119, 126, 129, 132, 135, 137, 146

GPT-4のベンチマーク結果

●ベンチマーク結果の取得

- ・ 「HumanEval on LLMs Revisted in Late 2023」 のデータから取得
- ・ GPT-4をHumanEval, pass@1でベンチマークした結果
- ・ スコア85.37（最大を100として正解した割合）

●pass@10の結果は取得できず

- ・ 論文ではpass@10でもベンチマークしたようだったが、データは取得できず
- ・ 自分で実行する場合リフレッシュしながら1640回の試行が必要になる
- ・ スコアが98を超えるので今回は必要性が低い

今後

●結果の確認

- ・ 分類によって正解分布に差が出るかどうか確かめ, 考察する

●他の分類

- ・ テストケースの数
- ・ プロンプトの長さ

●他のLLM