Большие языковые модели для генерации кода

Лебедев Андрей Алексеевич

Научный руководитель: Тихомиров Михаил Михайлович

МГУ им. М.В. Ломоносова, кафедра Алгоритмических языков

Проблемы

1. Увеличение спроса на программное обеспечение
2. Дефицит кадров среди разработчиков
3. Трудоемкость процесса разработки
4. Сложность изучения программирования

Актуальность

Автоматическая генерация кода позволяет решить следующие проблемы:

1. Автоматизация процесса программирования
2. Снижение затрат на разработку
3. Повышение доступности разработки
4. Улучшение качества кода

Постановка задачи

1. Исследовать большие языковые модели для генерации кода
2. Изучить способы оценки качества работы больших языковых моделей
3. Научиться развертывать, запускать и тестировать модели
4. Научиться работать с инструктивными и обычными моделями
5. Оценить качество работы некоторых моделей на русском языке по сравнению с английским

Какие задачи ставятся перед языковыми моделями для генерации кода?

1. Генерация
2. Завершение
3. Перевод на другие языки программирования
4. Модернизация
5. Обобщение
6. Тестирование и отладка

https://toloka.ai/blog/generative-ai-for-coding/

Способы оценки моделей генерации кода

Бенчмарки:

- MBPP
- HumanEval
- MultiPL-E

Метрика:

- pass@k

https://blog.continue.dev/an-introduction-to-code-llm-benchmarks-for-software-engineers/

MBPP

text string · *lengths* 82 104 30.7%	code string · *lengths* 294 382 7.8%	test_list sequence
Write a function to find the longest chain which can be formed from the given set of pairs.	class Pair(object): def __init__(self, a, b): self.a = a self.b = b def max_chain_length(arr, n): max = 0 mcl = [1 for i in range(n)] for i in range(1, n): for j in range(0, i): if (arr[i].a > arr[j].b and mcl[i] < mcl[j] + 1): mcl[i] = mcl[j] + 1 for i in range(n): if (max < mcl[i]): max = mcl[i] return max	["assert max_chain_length([Pair(5, 24), Pair(15, 25),Pair(27, 40), Pair(50, 60)], 4) == 3", "assert max_chain_length([Pair(1, 2), Pair(3, 4),Pair(5, 6), Pair(7, 8)], 4) == 4", "assert max_chain_length([Pair(19, 10), Pair(11, 12),Pair(13, 14), Pair(15, 16), Pair(31, 54)], 5) == 5"]

https://huggingface.co/datasets/mbpp

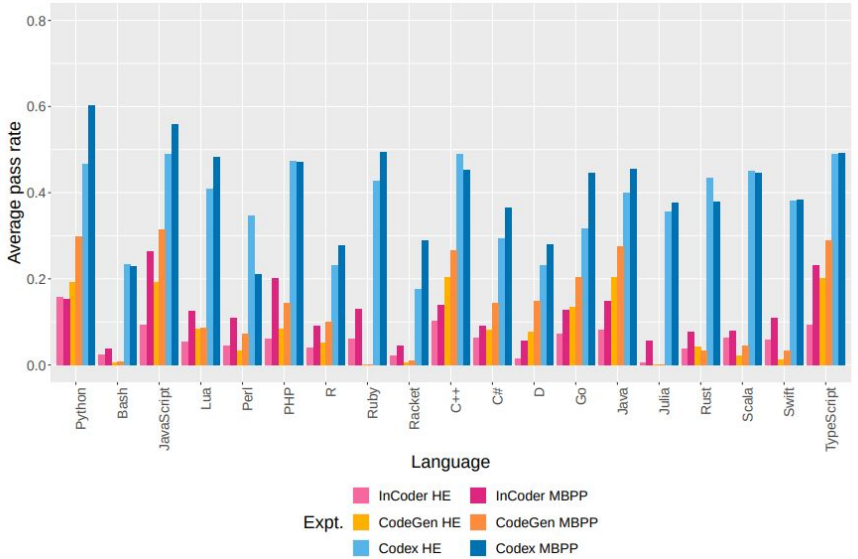
HumanEval

prompt string · *lengths*  240–365 24.4%	canonical_solution string · *lengths*  186–271 18.9%	test string · *lengths*  455–624 18.9%
`from typing import List def has_close_elements(numbers: List[float], threshold: float) -> bool: """ Check if in given list of numbers, are any two numbers closer to each other than given threshold. >>> has_close_elements([1.0, 2.0, 3.0], 0.5) False >>> has_close_elements([1.0, 2.8, 3.0, 4.0, 5.0, 2.0], 0.3) True """`	`for idx, elem in enumerate(numbers): for idx2, elem2 in enumerate(numbers): if idx != idx2: distance = abs(elem - elem2) if distance < threshold: return True return False`	`METADATA = { 'author': 'jt', 'dataset': 'test' } def check(candidate): assert candidate([1.0, 2.0, 3.9, 4.0, 5.0, 2.2], 0.3) == True assert candidate([1.0, 2.0, 3.9, 4.0, 5.0, 2.2], 0.05) == False assert candidate([1.0, 2.0, 5.9, 4.0, 5.0], 0.95) == True assert candidate([1.0, 2.0, 5.9, 4.0, 5.0], 0.8) == False assert candidate([1.0, 2.0, 3.0, 4.0, 5.0, 2.0], 0.1) == True assert candidate([1.1, 2.2, 3.1, 4.1, 5.1], 1.0) == True assert candidate([1.1, 2.2, 3.1, 4.1, 5.1], 0.5) == False`

https://huggingface.co/datasets/openai_humaneval

MultiPL-E

```cpp
// C++
#include<assert.h>
#include<bits/stdc++.h>
// You have been tasked to write a function that receives
// a hexadecimal number as a string and counts the number of hexadecimal
// digits that are primes (prime number, or a prime, is a natural number
// greater than 1 that is not a product of two smaller natural numbers).
// Hexadecimal digits are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.
// Prime numbers are 2, 3, 5, 7, 11, 13, 17,...
// So you have to determine a number of the following digits: 2, 3, 5, 7,
// B (=decimal 11), D (=decimal 13).
// Note: you may assume the input is always correct or empty string,
// and symbols A,B,C,D,E,F are always uppercase.
// Examples:
long hex_key(std::string num) {
  long sum = 0;
  for (int i = 0; i < num.length(); i++) {
    if (num[i] == '2' || num[i] == '3' || num[i] == '5'
     || num[i] == '7' || num[i] == 'B' || num[i] == 'D') {
      sum++; }
  }
  return sum;
}
```



https://arxiv.org/pdf/2208.08227

pass@k метрика

n – количество сгенерированных семплов
c – количество правильных

C – количество сочетаний $C_n^k = \frac{n!}{k!\,(n-k)!}$

$$pass@k := \mathbb{E}_{\text{problems}}\left[1 - \frac{C(n-c,k)}{C(n,k)}\right]$$

Цель метрики: оценить вероятность того, что по крайней мере одна из k лучших выборок является правильной.

https://deepgram.com/learn/humaneval-llm-benchmark

Обзор больших языковых моделей для генерации кода с открытым исходным кодом

1. DeepSeek Coder
2. CodeLlama
3. StarCoder2
4. WizardCoder

Инструктивные модели

Все модели предобучаются на большом количестве данных, во время этого процесса закладывается представление о мире

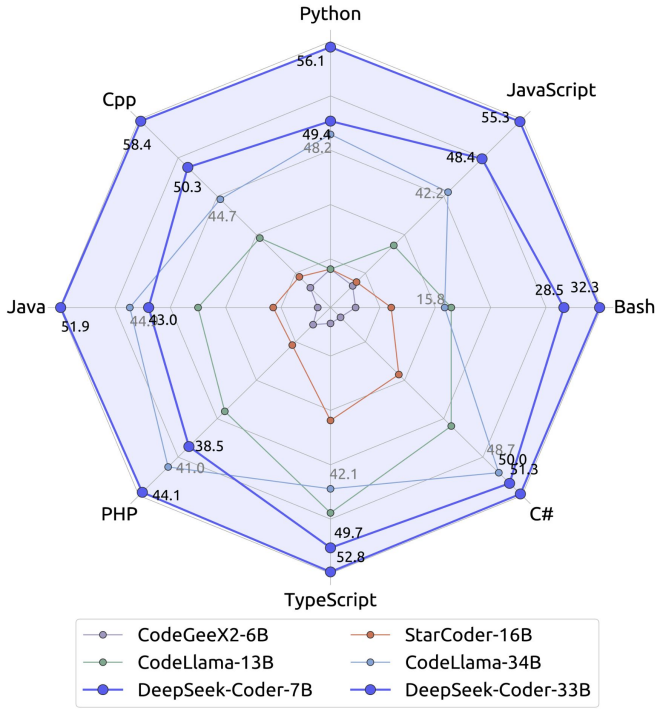
Инструктивные модели – модели, дообученные на задачу следования пользовательским инструкциями на естественном языке

DeepSeek Coder

- Предобучена на 2 триллионах токенов более чем на 80 языках программирования
- 87% кода и 13% данных на естественном языке (преимущественно на английском и китайском)
- Размеры: 1.3B, 5.7B, 6.7B и 33B
- Есть инструктивная версия
- Есть API

https://arxiv.org/pdf/2401.14196

DeepSeek Coder

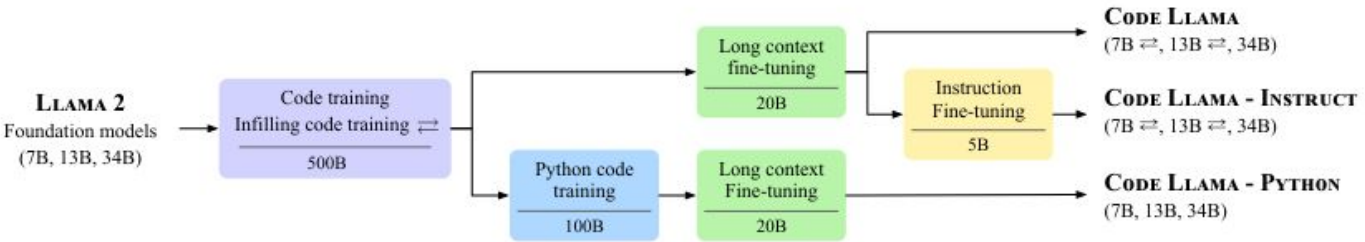


Model	Size	HumanEval		MBPP	DS-1000
		Python	Multilingual		
Pre-Trained Model					
Codex-001	-	33.5	26.1	45.9	20.2
Codex-002	-	-	-	-	39.2
CodeGeeX2	6B	36.0	24.5	42.4	22.9
StarCoder	16B	36.0	28.7	46.8	27.2
CodeLlama	7B	31.7	29.2	41.6	22.1
	13B	36.0	35.4	48.4	26.8
	34B	48.2	41.0	55.2	34.3
DeepSeek-Coder Base	1B	34.8	28.3	46.2	16.2
	7B	49.4	44.7	60.6	30.5
	33B	56.1	50.3	66.0	40.2
Instruction-Tuned Model					
GPT-3.5-turbo	-	76.2	64.9	70.8	-
GPT4	-	84.1	76.5	80.0	-
DeepSeek-Coder Instruct	7B	78.6	66.1	65.4	-
	33B	79.3	69.2	70.0	-

https://arxiv.org/pdf/2401.14196

CodeLlama

- Построена на базе Llama2
- Размеры: 7B, 13B, 34B
- Хорошо решает задачи завершения и заполнения кода



https://arxiv.org/pdf/2308.12950

CodeLlama

Model	Size	HumanEval			MBPP		
		pass@1	pass@10	pass@100	pass@1	pass@10	pass@100
code-cushman-001	12B	33.5%	-	-	45.9%	-	-
GPT-3.5 (ChatGPT)	-	48.1%	-	-	52.2%	-	-
GPT-4	-	67.0%	-	-	-	-	-
PaLM	540B	26.2%	-	-	36.8%	-	-
PaLM-Coder	540B	35.9%	-	88.4%	47.0%	-	-
PaLM 2-S	-	37.6%	-	88.4%	50.0%	-	-
StarCoder Base	15.5B	30.4%	-	-	49.0%	-	-
StarCoder Python	15.5B	33.6%	-	-	52.7%	-	-
StarCoder Prompted	15.5B	40.8%	-	-	49.5%	-	-
Llama 2	7B	12.2%	25.2%	44.4%	20.8%	41.8%	65.5%
	13B	20.1%	34.8%	61.2%	27.6%	48.1%	69.5%
	34B	22.6%	47.0%	79.5%	33.8%	56.9%	77.6%
	70B	30.5%	59.4%	87.0%	45.4%	66.2%	83.1%
Code Llama	7B	33.5%	59.6%	85.9%	41.4%	66.7%	82.5%
	13B	36.0%	69.4%	89.8%	47.0%	71.7%	87.1%
	34B	48.8%	76.8%	93.0%	55.0%	76.2%	86.6%
	70B	53.0%	84.6%	96.2%	62.4%	81.1%	91.9%
Code Llama - Instruct	7B	34.8%	64.3%	88.1%	44.4%	65.4%	76.8%
	13B	42.7%	71.6%	91.6%	49.4%	71.2%	84.1%
	34B	41.5%	77.2%	93.5%	57.0%	74.6%	85.4%
	70B	67.8%	90.3%	97.3%	62.2%	79.6%	89.2%
Unnatural Code Llama	34B	62.2%	85.2%	95.4%	61.2%	76.6%	86.7%
Code Llama - Python	7B	38.4%	70.3%	90.6%	47.6%	70.3%	84.8%
	13B	43.3%	77.4%	94.1%	49.0%	74.0%	87.6%
	34B	53.7%	82.8%	94.7%	56.2%	76.4%	88.2%
	70B	57.3%	89.3%	98.4%	65.6%	81.5%	91.9%

https://arxiv.org/pdf/2308.12950

StarCoder2

- Предобучена на более чем 4 триллионах токенов и более чем 600 языках программирования
- Обучалась на задачу заполнения кода
- Размеры: 3B, 7B и 15B

https://arxiv.org/pdf/2402.19173

StarCoder2

Model	HumanEval	HumanEval+	MBPP	MBPP+
StarCoderBase-3B	21.3	17.1	42.6	35.8
DeepSeekCoder-1.3B	28.7	23.8	55.4	46.9
StableCode-3B	28.7	24.4	53.1	43.1
StarCoder2-3B	**31.7**	**27.4**	**57.4**	**47.4**
StarCoderBase-7B	30.5	25.0	47.4	39.6
CodeLlama-7B	33.5	25.6	52.1	41.6
DeepSeekCoder-6.7B	**47.6**	**39.6**	**70.2**	**56.6**
StarCoder2-7B	35.4	29.9	54.4	45.6
StarCoderBase-15B	29.3	25.6	50.6	43.6
CodeLlama-13B	37.8	32.3	62.4	52.4
StarCoder2-15B	**46.3**	**37.8**	**66.2**	**53.1**
CodeLlama-34B	48.2	44.3	65.4	52.4
DeepSeekCoder-33B	**54.3**	**46.3**	**73.2**	**59.1**

https://arxiv.org/pdf/2402.19173

WizardCoder

- Построена на базе CodeLlama
- Размеры: 7B, 13B, 70B
- Использован Evol-Instruct метод
- Evol-Instruct — это метод, использующий языковые модели (ChatGPT 3.5) вместо людей для автоматического написания инструкций с открытым доменом различных уровней сложности и диапазона навыков.

https://arxiv.org/pdf/2306.08568

WizardCoder

Model	Params	HumanEval	MBPP
Closed-source models			
LaMDA [40]	137B	14.0	-
AlphaCode [12]	1.1B	17.1	-
PaLM [3]	540B	26.2	36.8
PaLM-Coder [3]	540B	36.0	47.0
PaLM 2-S [4]	-	37.6	50.0
Codex [16]	2.5B	21.4	-
Codex [16]	12B	28.8	-
Code-Cushman-001 [38]	-	33.5	45.9
Code-Davinci-002 [38]	-	47.0	58.1
GPT-3.5 [2]	-	48.1	-
GPT-4 [2]	-	67.0	-
Open-source models			
LLaMa [8]	33B	21.7	30.2
LLaMa [8]	65B	23.7	37.7
CodeGen-Multi [13]	16B	18.3	20.9
CodeGen-Mono [13]	16B	29.3	35.3
CodeGeeX [14]	13B	22.9	24.4
StarCoder [11]	15B	33.6	43.6*
CodeT5+ [18]	16B	30.9	-
InstructCodeT5+ [18]	16B	35.0	-
WizardCoder	15B	**57.3** (+22.3)	**51.8** (+8.2)

https://arxiv.org/pdf/2306.08568

Эксперименты

Промптинг: zero-shot

< | begin_of_sentence | > You are a smart assistant in writing code that helps the user solve his tasks. Below is an instruction describing the task. Write an answer that exactly fulfills the user's request.

Instruction:

Write a python function to remove first and last occurrence of a given character from the string. The function should have the following name: remove_Occ.

Response:

Промптинг: few-shot

< | begin_of_sentence | > You are a smart assistant in writing code that helps the user solve his tasks. Below is an instruction describing the task. Write an answer that exactly fulfills the user's request.

Instruction:
Write a function to find the minimum cost path to reach (m, n) from (0, 0) for the given cost matrix cost[][] and a position (m, n) in cost[][]. The function should have the following name: min_cost.
Response:
expected model's answer
<|EOT|>

Instruction:
Write a function to find the similar elements from the given two tuple lists. The function should have the following name: similar_elements.
Response:
expected model's answer
<|EOT|>

Instruction:
Write a python function to remove first and last occurrence of a given character from the string. The function should have the following name: remove_Occ.
Response:

DeepSeek 7B Instruct MBPP zero-shot

Задача: протестировать инструктивную модель DeepSeek 7B на бенчмарке MBPP с системным промптом, но без подсказок

Цели:

1. Научиться запускать и оценивать модель
2. Изучить влияние добавления системного промпта
3. Подобрать лучшие параметры и изучить поведение модели при их изменении
4. Посмотреть на качество модели на задачах на естественном языке

Результат: 34,6% (173/500 тестов)

DeepSeek 7B Instruct MBPP few-shot

Задача: протестировать инструктивную модель DeepSeek 7B на бенчмарке MBPP с системным промптом и с подсказками (few-shot)

Цель: рассмотреть, как изменение промпта путем добавления примеров диалогов между пользователем и моделью может повлиять на качество генерации

Результат: 38,2% (191/500 тестов)

DeepSeek 7B Instruct EN HumanEval zero-shot

Задача: протестировать инструктивную модель DeepSeek 7B на бенчмарке HumanEval с системным промптом, но без подсказок

Цели:

1. Посмотреть, как изменится качество генерации, если решается задача продолжения кода вместе задачи на естественном языке (как это было в MBPP)
2. Оценить качество модели на бенчмарке HumanEval

Результат: 68,3% (112/164 тестов)

RU HumanEval

docstring string · *lengths* 179–302 24.4%	prompt string · *lengths* 245–372 23.8%	canonical_solution string · *lengths* 186–271 18.9%	test string · *lengths* 455–624 18.9%
Проверьте, есть ли в заданном списке чисел какие-либо два числа ближе друг к другу, чем заданный порог. >>> has_close_elements([1.0, 2.0, 3.0], 0.5) False >>> has_close_elements([1.0, 2.8, 3.0, 4.0, 5.0, 2.0], 0.3) True	from typing import List def has_close_elements(numbers: List[float], threshold: float) -> bool: """ Проверьте, есть ли в заданном списке чисел какие-либо два числа ближе друг к другу, чем заданный порог. >>> has_close_elements([1.0, 2.0, 3.0], 0.5) False >>> has_close_elements([1.0, 2.8, 3.0, 4.0, 5.0, 2.0], 0.3) True """	for idx, elem in enumerate(numbers): for idx2, elem2 in enumerate(numbers): if idx != idx2: distance = abs(elem - elem2) if distance < threshold: return True return False	METADATA = { 'author': 'jt', 'dataset': 'test' } def check(candidate): assert candidate([1.0, 2.0, 3.9, 4.0, 5.0, 2.2], 0.3) == True assert candidate([1.0, 2.0, 3.9, 4.0, 5.0, 2.2], 0.05) == False assert candidate([1.0, 2.0, 5.9, 4.0, 5.0], 0.95) == True assert candidate([1.0, 2.0, 5.9, 4.0, 5.0], 0.8) == False assert candidate([1.0, 2.0, 3.0, 4.0, 5.0, 2.0], 0.1) == True assert candidate([1.1, 2.2, 3.1, 4.1, 5.1], 1.0) == True assert candidate([1.1, 2.2, 3.1, 4.1, 5.1], 0.5) == False

https://huggingface.co/datasets/NLPCoreTeam/humaneval_ru/

DeepSeek 7B Instruct RU HumanEval zero-shot

Задача: протестировать инструктивную модель DeepSeek 7B на бенчмарке RU HumanEval с системным промптом, но без подсказок

Цель: оценить, как изменится качество генерации на тех же заданиях, что в оригинальном MBPP, но сформулированных на русском языке

Результат: 67,6% (111/164 тестов)

StarCoder2 7B HumanEval, RU HumanEval

Задача: протестировать не инструктивную модель StarCoder2 7B на бенчмарках HumanEval и RU HumanEval с системным промптом

Цели:

1. Оценить качество генерации модели StarCoder2 на бенчмарках, где задача состоит в продолжении кода
2. Оценить поведение модели на русском языке по сравнению с английским

Текущие результаты

В рамках курсовой работы удалось:

1. Научиться запускать большие языковые модели генерации кода с открытым исходным кодом, оценивать их качество
2. Научиться работать с инструктивными моделями
3. Протестировать наиболее популярные модели генерации кода на бенчмарках HumanEval и MBPP
4. Сравнить поведение модели при работе с русским и английским языками
5. Воспроизвести результаты модели DeepSeek 7B Instruct на датасете HumanEval как в оригинальной статье

Текущие результаты

Были сделаны следующие выводы:

1. few-shot для инструктивных моделей помогает повысить качество генерации
2. В целом модели устойчивы к отклонениям гиперпараметров от оптимальных, при этом параметры по умолчанию таковыми не являются, то есть их стоит подбирать под определенную задачу
3. Качество модели не ухудшается при работе на русском языке по сравнению с английским
4. Актуальные модели с открытым исходным кодом и небольшим количеством параметров достаточно плохо справляются с решением задач, представленных на естественном языке

Дальнейшие планы

1. Оценить, как квантизация модели влияет на качество ее работы
2. Оценить качество других, более крупных, моделей генерации кода
3. Дообучить модель на конкретную задачу

Источники

1. Ziyang Luo и др. “WizardCoder: Empowering Code Large Language Models with Evol-Instruct”. В: arXiv:2306.08568v1 (2023)
2. Xiao Bi и др. “DeepSeek LLM. Scaling Open-Source Language Models with Longtermism”. В: arXiv:2401.02954v1 (2024)
3. Anton Lozhkov и др. “StarCoder 2 and The Stack v2: The Next Generation”. В: arXiv:2402.19173v1 (2024)
4. Baptiste Rozière и др. “Code Llama: Open Foundation Models for Code”. В: arXiv:2308.12950v3 (2024)
5. Mark Chen и др. “Evaluating Large Language Models Trained on Code”. В: arXiv:2107.03374v2 (2021)
6. Maxwell Nye и др. “Program Synthesis with Large Language Models”. В: arXiv:2108.07732v1 (2021)