

Code Llama, a state-of-the-art large language model for coding

Danh Thien Luu (79663)

Inhaltsverzeichnis

1	Einführung	1
2	Über Code Llama	2
3	Einrichtung von Code Llama	4
4	Durchführung	6
4.1	Generierung von Code	6
5	Vergleich mit anderen Chatbots	8
6	Ergebnisse	9
7	Fazit	10
8	Referenzen	11

1 Einführung

Code Llama ist ein neues State of the Art Sprachmodell, spezialisiert zur Generierung von Code und natürlicher Sprache über Code. Dazu akzeptiert sie sowohl Prompts, die Code enthalten als auch welche, die natürliche Sprache enthalten.

Entwickelt wurde sie von Meta, der Muttergesellschaft von Facebook und ist frei zu Forschungs- und Kommerziellen Zwecken nutzbar. Veröffentlicht wurde Code Llama am 24. August und ist, ein auf Programmiercode spezialisierte Version von Llama 2, welches durch ein erweitertes, längeres Training des bestehenden Code-Datensatzes entstand. [2]

Als Ergebnis, kann Code Llama Code generieren, und unterstützt zudem

In dieser Projektarbeit wird Code Llama auf die beworbenen Fähigkeiten getestet und auf die Nutzbarkeit in der echten Welt geprüft.

2 Über Code Llama

Code Llama gibt es in verschiedenen Varianten:

- Default
- Instruct
- Python
- Unnatural

Diese unterscheiden sich in erster Linie durch den Datensatz, die zum Trainieren verwendet wurde.

Die Default-Variante ist die Standardvariante, die auf dem Datensatz, die vorher beschrieben wurde, basiert.

Bei der Python-Variante wurde mehr Python-Code zum feintunen verwendet, somit kann dieses Modell in der Theorie besser mit Python-Code umgehen.

Bei der Instruct Variante handelt es sich um ein Modell, welches mit menschlichen Instruktions-Datensätzen fein getunt worden ist. Man nennt dies dann aligned, dies bedeutet die Ausgabe des Modells ist konsistent zu dem, wie es ein Mensch erwarten würde und ermöglicht es dem Modell, z.B. auf Fragen zu antworten oder andere menschenähnliche Interaktionen zu erzeugen. Somit ist das die nutzerfreundlichste Version von Code Llama.

Es gibt auch noch eine Variante, die als Unnatural Code Llama bezeichnet wird. Diese Version wird der Öffentlichkeit leider (noch) nicht zur Verfügung gestellt. Sie schneidet im Vergleich zu den anderen Sprachmodellen, die im Research Paper verglichen wurden in allen bis auf einer Rubrik am besten ab und dürfte somit Metas mächtigstes Sprachmodell für Code sein. Erschaffen wurde es, indem Code Llama - Python anhand von 15.000 unnatürlichen Instruktionen feingetunt worden ist, also ein Datensatz, der vollkommen synthetisch und automatisiert mithilfe von anderen Sprachmodellen erzeugt wurde.

Diese verschiedenen Variationen gibt es dann nochmal in der 7b, 13b und der 34b Version. Diese unterscheiden sich vor allem in der Größe des Sprachmodells oder um genau zu sein wurden diese jeweils mit 7 Milliarden, 13 Milliarden und 34 Milliarden verschiedenen Parametern trainiert.

Damit dürfte die 34b Version die besten Ergebnisse liefern während die 7b und 13b Versionen schneller sind und sich daher eher für Echtzeit Code-Completion eignen.

Es gibt viele Sprachmodelle, oder genauer bezeichnet „Large Language Models“ (LLM), die in „Konkurrenz“ mit Code Llama stehen wie z.B. GPT-4, die im selben oder auch anderen Bereichen besser performen können. Um das auszuwerten gibt es verschiedene Kennzahlen, um diese Performance messen.

In der Tabelle, die im Research Paper auftaucht, werden verschiedene Sprachmodelle miteinander anhand von verschiedenen Metriken verglichen. Je höher die Zahl in der Kategorie, desto besser hat dieses Sprachmodell in der Evaluation performt.

Code Llama - Python 34b hat beispielsweise bei der Pass@1 HumanEval Kategorie ein Ergebnis von 53,7% erreicht, während GPT-4, das beste Modell in dieser Kategorie, 67% erreicht hat. Das ist in anbetracht der Größe der jeweiligen Modelle äußerst bemerkenswert, da Code Llama nur mit einem Bruchteil der Größe von GPT-4 (mindestens 1 Billion Parameter) bereits ähnlich gute Ergebnisse liefert.

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%
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%
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%

Abbildung 1: Code LLama im Vergleich zu anderen LLMs [4]

3 Einrichtung von Code Llama

Um Code Llama zu benutzen gibt es verschiedene Möglichkeiten.

Die erste Möglichkeit wäre die Einrichtung von direkt aus dem Quellcode von Code Llama lokal auf z.B. dem eigenem Rechner. In der GitHub Repository von Code Llama [1] gibt es eine etwas detailliertere Anleitung wie dies funktioniert. Die Vorteile von solch einer lokalen Installation ist die Unabhängigkeit von externen Servern, Offline nutzbarkeit und der damit verbundene Datenschutz. Jedoch läuft es nicht auf jedem Rechner gut aufgrund von hohen Systemanforderungen. Neben einem hohen Speicherbedarf (sowohl Festplatte als auch Arbeitsspeicher) ist auch ein performanter Prozessor nötig, um schnelle Ergebnisse zu erzielen.

Eine alternative und einfachere Methode, um schnell Code Llama benutzen zu können sind Chatbots welche besonders Nutzerfreundlich sind, da man keine Einrichtung benötigt und man direkt mit der Eingabe von Prompts in einer intuitiven UI beginnen kann. Perplexity Labs [3] bietet u.a. einen Chatbot mit dem Code Llama 34b Instruct Modell an

Aufgrund von Hardware Limitierung wird daher der angebotene Chatbot mit Code Llama genutzt, um es in dieser Projektarbeit zu testen.

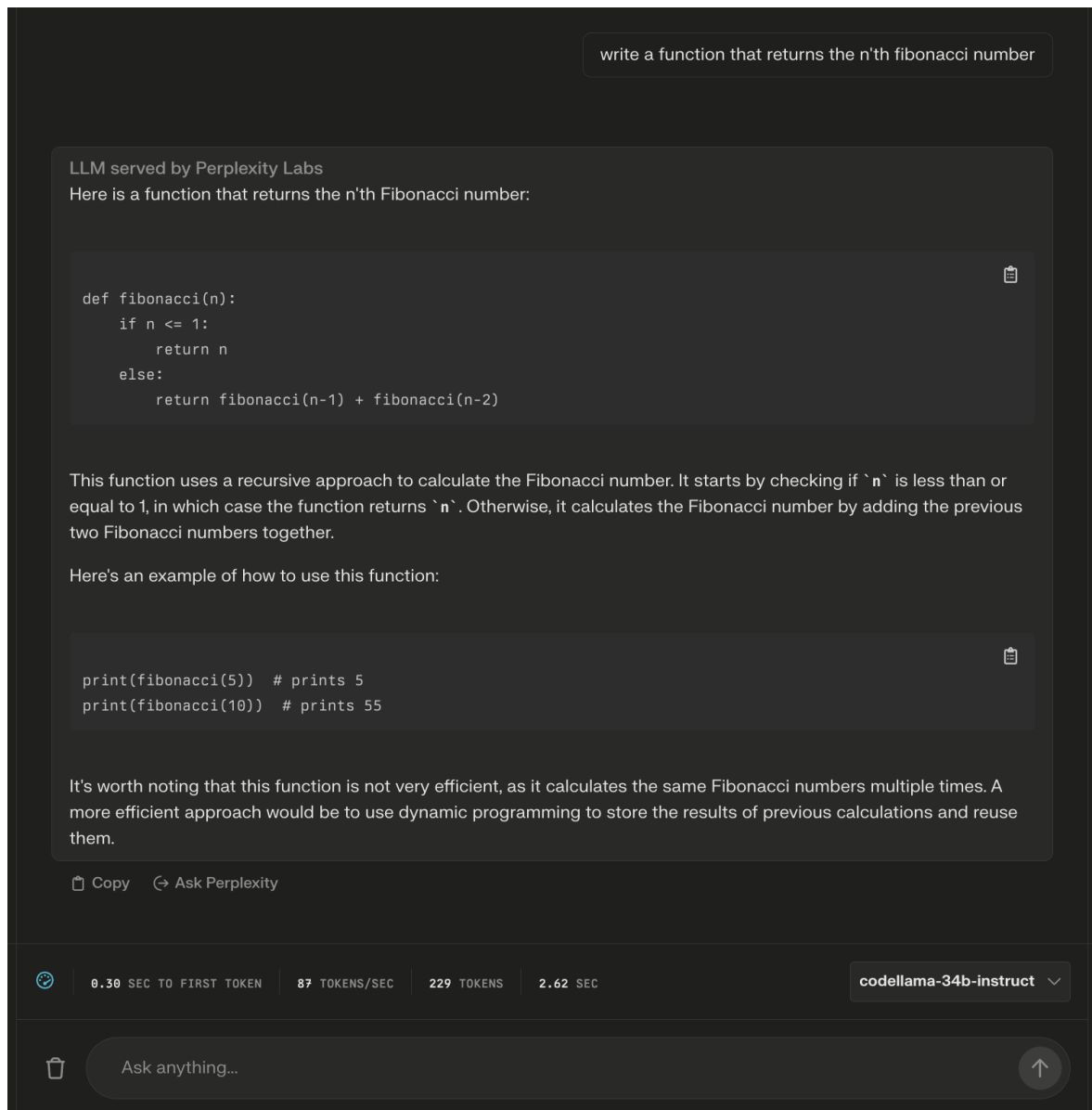


Abbildung 2: Code Llama im Perplexity Labs Chatbot

4 Durchführung

Um die Fähigkeiten der Code- und Sprachgenerierung von Code Llama auszuprobieren und zu testen, könnten folgende Usecases Interessant sein:

- Generierung von Code anhand einer Beschreibung in natürlicher Sprache
- Code Reviews
- Dokumentation
- Verbessern von bestehendem Code (Refactoring)
- Zusammenfassung von Code
- Potenzielle Bugs erkennen und Hilfestellung bei Debugging leisten

4.1 Generierung von Code

Der wohl am naheliegende Use-case ist die Generierung von Code anhand eines Prompts in natürlicher Sprache.

Folgende Ergebnisse wurden erzielt:

Prompt:

Generiere Code

Antwort:

Here is a function that returns the n'th Fibonacci number:

```
def fibonacci(n):  
    if n <= 1:  
        return n  
    else:  
        return fibonacci(n-1) + fibonacci(n-2)
```

This function uses a recursive approach to calculate the Fibonacci number.

It starts by checking if n is less than or equal to 1, in which case the function returns n.

Otherwise, it calculates the Fibonacci number by adding the previous two Fibonacci numbers together.

Here's an example of how to use this function:

```
print(fibonacci(5)) # prints 5
```

```
print(fibonacci(10)) # prints 55
```

It's worth noting that this function is not very efficient, as it calculates the same Fibonacci numbers multiple times. A more efficient approach would be to use dynamic programming to store the results of previous calculations and reuse them.

5 Vergleich mit anderen Chatbots

6 Ergebnisse

7 Fazit

8 Referenzen

- [1] 2023. GitHub - Codellama. Abgerufen 20. Oktober 2023 von <https://github.com/facebookresearch/codellama>
- [2] Introducing Code Llama, a State-of-the-Art Large Language Model for Coding. Abgerufen 16. Oktober 2023 von <https://ai.meta.com/blog/code-llama-large-language-model-coding/>
- [3] Perplexity Labs. Abgerufen 2. Dezember 2023 von <https://labs.perplexity.ai/>
- [4] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, und Gabriel Synnaeve. Code Llama: Open Foundation Models for Code.