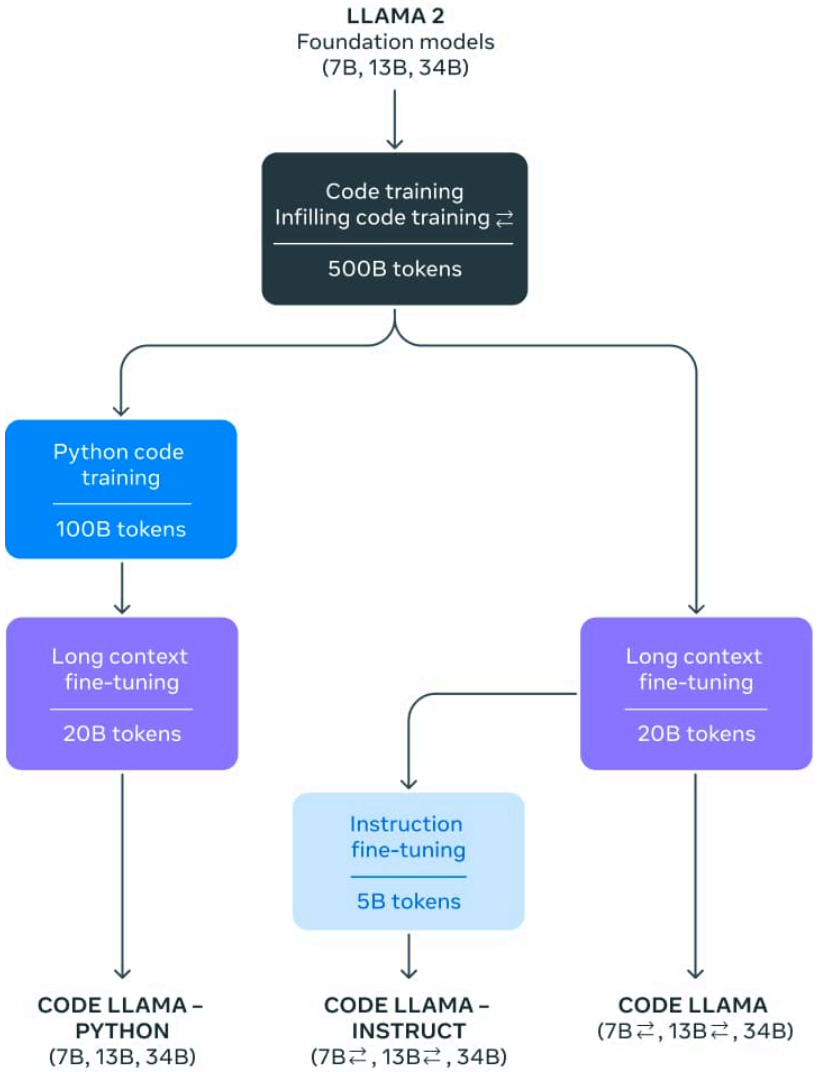
LLM для генерации кода

5 моделей с открытым исходным кодом

1. **CodeLlama**

   Есть 3 варианта модели:
   1) Классическая CodeLlama
   2) Дообученная на Python
   3) Дообученная на задачах на естественном языке



   Каждый из вариантов представлен в 3 размерах: 7, 13 и 34 миллиарда параметров.

Датасет неизвестен. Результаты бенчмарков можно посмотреть здесь: https://huggingface.co/spaces/bigcode/bigcode-models-leaderboard.

2. **Phind-CodeLlama**

   Первая версия основана на CodeLlama 34B и настроена для диалога на естественном языке.
   Все, что известно о датасете:
   We fined-tuned on a proprietary dataset of 1.5B tokens of high quality programming problems and solutions. This dataset consists of instruction-answer pairs instead of code completion examples, making it structurally different from HumanEval. LoRA was not used -- both models are a native finetune. We used DeepSpeed ZeRO 3 and Flash Attention 2 to train these models in 15 hours on 32 A100-80GB GPUs. We used a sequence length of 4096 tokens.

3. **WizardCoder-Python**

   https://towardsai.net/p/machine-learning/wizardcoder-why-its-the-best-coding-model-out-there – здесь пишут, что отличительной особенностью WizardCoder является датасет, где было сбалансировано количество простых и сложных инструкций. Таким образом, модель лучше работает с языком и, соответственно, лучше проходит humaneval.

4. **DeepSeek Coder**

   Модель была обучена с нуля на наборе данных, включающем 87% кода и 13% естественного языка на английском и китайском. Также имеет 3 размера: от 1.3 до 33 миллиардов параметров. Одной из отличительных особенностей является поддержка написания и заполнения кода на уровне проекта благодаря предварительной подготовке на корпусе кодов на уровне проекта.

5. **StarCoder**

   15 миллиардов параметров. Обучалась на кодах с github. Контекстное окно 8192 токена, что превосходит конкурентов. К особенностям можно отнести, во-первых, что модель можно использовать как технического ассистента, а во-вторых, насколько я понял, ее способность отлаживать код.

Задачи, которые решают модели, генерирующие код

1. Упрощение работы разработчиков. Эта задача ставилась перед всеми моделями, она помогает программистам сосредоточиться на более сложных вещах и отдать модели написание множества простых блоков кода, функций. Сложные и крупные уже получаются некачественными, хотя, как я писал выше, DeepSeek coder теоретически может и такое.

2. Написание тестов и отладка – то, что мы с Вами обсуждали. Написание тестов – очень распространенная проблема в алгоритмических задачах. А вот отлаживать код модели не умеют, насколько я понимаю (за исключением теоретической такой возможности у StarCoder). Кстати, для меня это вопрос, почему GPT 3.5 при задаче написать код просто склеивает его как обычное предложение, а не предварительно тестирует и выдает отлаженный результат.

3. Такой задачи еще не ставилось, но она кажется мне интересной: написать модель, которая будет обучать программированию. У нее немного иначе должно строиться взаимодействие. То есть она должна мочь рассказать какую-то тему, дать человеку подсказку, указать на ошибки в коде и тд, это уже методики обучения.