Локальный inference крупных языковых моделей: производительность, память и применимость

Ниязова Барнонисо Наимджоновна
аспирант 4 курса,
Российский государственный университет имени Косыгина,
г. Москва

Аннотация: Цель данной статьи — провести сравнительный анализ производительности локального инференса крупных языковых моделей, оценить их применимость в задачах пользовательского интерфейса (UI) и автоматического тестирования, а также сформулировать рекомендации по выбору архитектур для запуска без подключения к внешним API. В основе анализа лежит опыт интеграции модели DeepSeek-R1 в систему генерации тест-кейсов UI/API на базе Playwright, с использованием среды MacBook на процессоре Apple M1 с поддержкой Metal Performance Shaders (MPS).

Ключевые слова: тестирование, инструменты тестирования, информационные технологии, веб-приложения.

В последние годы крупные языковые модели (Large Language Models, LLMs) стали краеугольным камнем современных систем искусственного интеллекта. Такие модели, как GPT-4, LLaMA, DeepSeek и их производные, демонстрируют высокую эффективность в задачах генерации, анализа текста, программирования и даже принятия решений. Однако широкое распространение LLM сопряжено с рядом вызовов, среди которых ключевыми являются: производительность при локальном запуске, требования к аппаратным ресурсам, ограничения памяти и применимость в прикладных задачах.

Обзор архитектур и требований

Современные языковые модели отличаются по следующим параметрам:

— Объём модели (число параметров): от 1B до 70B+

— Формат загрузки (FP32, BF16, INT4/8)

— Максимальная длина контекста

— Архитектура: Dense vs MoE (Mixture of Experts)

— Поддержка MPS, CUDA, ROCm, CPU

Таблица 1. Сравнение моделей по требованиям

Модель	Параметров	Формат	Память (RAM/VRAM)	Контекст	Поддержка MPS	Примерная скорость
GPT-2 Large	762M	FP32	~4 ГБ	1024	Да	12–15 токенов/с
DeepSeek-R1-Distill-Qwen-1.5B	1.5B	INT4	~3 ГБ	8K	Частично	10 токенов/с
LLaMA 3 8B	8B	BF16	~16 ГБ	16K	Да (через vLLM)	5–8 токенов/с
DeepSeek-R1	67B	FP32	80–120 ГБ	128K	Нет	

Методика оценки

Для оценки производительности инференса была реализована система генерации шагов тест-кейсов с использованием LLM. Архитектура включает:

— Frontend (HTML) интерфейс для отображения плана тестирования

— Backend на FastAPI с маршрутом /generate-plan, обрабатывающим URL

— Модуль генерации generate_with_deepseek(prompt: str) для обращения к модели

— Вывод результатов в PDF/Excel и в UI

Результаты экспериментов

Эксперименты проводились на MacBook Pro с M1 Pro (32 ГБ RAM). В качестве модели использовался DeepSeek-R1-Distill-Qwen-1.5B, загруженный через transformers и AutoModelForCausalLM.

Таблица 2. Сравнение режимов

Режим генерации	Средняя задержка (сек)	Размер prompt	Использование памяти
API HuggingFace	~3.2	150 токенов	—
Локально (MPS)	~1.1	150 токенов	~2.5 ГБ
CUDA (десктоп)	~0.4	150 токенов	~1.8 ГБ

Пример запроса и генерации

Prompt: «Напиши подробные шаги и ожидания для теста на проверку клика по ссылке с текстом: ‘Контакты’»

Ответ (сгенерировано локально):

Шаг 1 — открыть главную страницу сайта

Шаг 2 — найти элемент «Контакты»

Шаг 3 — кликнуть по элементу

Шаг 4 — проверить, что URL изменился на «/contacts»

Шаг 5 — убедиться, что отображается заголовок «Контакты»

Ожидание: пользователь должен увидеть страницу с контактной информацией

Анализ применимости и ограничений

Когда локальный запуск оправдан

Локальный инференс крупных языковых моделей становится особенно актуален в следующих случаях:

1. Ограничение доступа к API: например, при разработке в условиях, когда HuggingFace или OpenAI API недоступны из-за географических или сетевых ограничений.

2. Снижение затрат: коммерческое использование API в высоконагруженных системах может обходиться в десятки тысяч рублей в месяц. Локальный запуск (единовременная настройка модели на сервере или ноутбуке) позволяет сократить расходы.

3. Повышенные требования к конфиденциальности: при обработке чувствительных данных (например, генерирование тестов по внутренним CRM или HR-системам) локальный запуск исключает утечку данных на внешние сервера.

4. Воспроизводимость результатов: в ряде исследований важно сохранять контроль над моделью, версией, seed и архитектурой. Это возможно только при локальном инференсе.

Ограничения локального запуска

Несмотря на плюсы, есть и ряд ограничений:

— Задержка старта (cold start): запуск моделей занимает до 30 секунд в зависимости от размера модели и количества VRAM.

— Поддержка железа: даже при наличии поддержки Metal (MPS) на macOS, не все модели могут эффективно работать. Например, некоторые архитектуры не оптимизированы под M1/M2.

— Ограничения по длине контекста: многие модели (например, GPT-2 или Qwen 1.5B) поддерживают контекст только до 2048–4096 токенов, что ограничивает применимость для анализа больших UI-страниц или API-спецификаций.

Выводы

Локальный запуск LLM позволяет создать независимую, автономную систему генерации и валидации UI/API тестов. Он критически важен:

— в закрытых корпоративных сетях,

— при работе с NDA-проектами,

— для оптимизации затрат.

Тем не менее, локальный inference требует точной настройки, знания доступных архитектур (Qwen, DeepSeek, LLaMA) и наличия хотя бы 16–32 ГБ оперативной памяти или GPU.

Главное о локальном инференсе

— Локальный запуск моделей работает — и стабильно на M1/M2, но только с оптимизированными версиями (Qwen, DeepSeek Distill, GPT-2/3).

— DeepSeek-R1-Distill-Qwen-1.5B — хороший баланс: <4 ГБ памяти, быстрый ответ, поддержка сложных шагов.

— Интеграция с тестирующим фреймворком Playwright позволяет автоматизировать UI-покрытие.

— Необходима проверка: не все модели поддерживают корректную генерацию шагов/ожиданий без fine-tuning.

— Формулы расчёта ресурсоёмкости и скорости позволяют подобрать правильную модель под железо.

Список источников:

1. Цзи-Цай Ян, Цзюнь-Лун Хуан, Фэн-Цзянь Ван, Уильям и Ч. Чу. Построение объектно-ориентированной архитектуры для тестирования веб-приложений. Журнал информационных наук и техники, январь 2016.

2. Цзи-Цай Ян, Цзюнь-Лун Хуан, Фэн-Цзянь Ван и Уильям К. Чу. Объектно-ориентированная архитектура, поддерживающая тестирование веб-приложений. Материалы 23-й Международной конференции по компьютерному программному обеспечению и приложениям, Конференция по катионам (COMPSAC ’99), страницы 122-127. Компьютер IEEE Общество, 2014.

3. Филиппо Рикка и Паоло Тонелла. Анализ веб-сайта: Структура и эволюция. В материалах 16-й Международной конференции по программному обеспечению техническое обслуживание (ICSM ’00), страницы 76-87. Компьютерное общество IEEE, 2017

4. Филиппо Рикка и Паоло Тонелла. Анализ и тестирование веб-приложений. В материалах 23-й Международной конференции по программному обеспечению и инжиниринг (ICSE ’01), страницы 25-34. Компьютерное общество IEEE, 2010

5. Филиппо Рикка и Паоло Тонелла. Процессы тестирования веб-приложений. Анналы разработки программного обеспечения, 14:93-114, декабрь 2009 года.

6. Майкл Бенедикт, Джулиана Фрейре и Патрис Годфруа. Veriweb: В материалах 11-й Международной конференции по Всемирной паутине (WWW ’02), Гонолулу, США, 2009.

7. Себастьян Эльбаум, Срикант Карре и Грегг Ротермел. В ходе разбирательств 25-я Международной конференции по разработке программного обеспечения (ICSE ’03), страницы 49-59. Компьютерное общество IEEE, 2008.