

Основы практического применения больших языковых моделей (LLM) в научно-технической деятельности

Презентацию подготовил Кудрявцев Виктор (с небольшой помощью LLM :))

Введение

- 1) Техническое введение (Что такое LLM)
- 2) Подготовка и настройка доступа (развертывание LLM)
- 3) Применение LLM
- 4) Короткое How To

Что такое LLM?

- Большая языковая модель – это статистическая модель, обученная на огромных объемах текстовых данных, способная предсказывать следующее слово (токен) в последовательности.
- Это не разум, а очень сложный автокомплит.
- Промт:
"Напиши короткий стишок про атом."
- Ответ LLM:
"Маленький атом, крутится, вертится,
Вокруг ядра электрон мчится,
Энергию он излучает,
Мир науки озаряет!"

Немного истории

- **Предыстория и другие методы:**

До-нейронные сети:

- Правила, статистические методы.

Рекуррентные нейронные сети (RNN), LSTM, GRU:

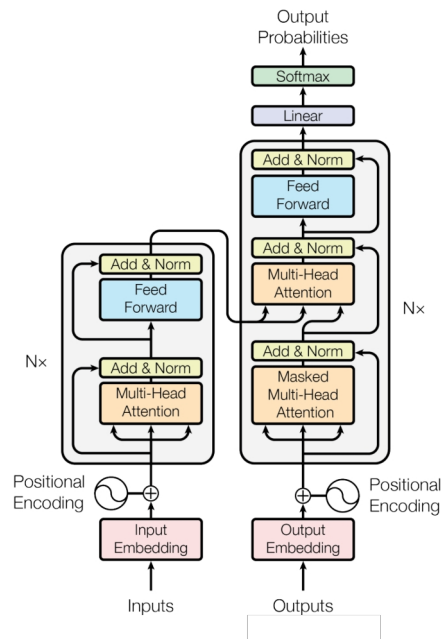
- Модели для последовательных данных.
- Проблема: "забывали" контекст на длинных последовательностях, сложно параллелились.

Ключевые концепции

- Трансформеры (Transformer Architecture):
- Токенизатор - компонент, который разбивает текст на более мелкие единицы – токены (слова, части слов, символы).

BERT

Encoder



GPT

Decoder

Параметры LLM

- **Число параметров:**
 - Количество весов в нейронной сети (миллиарды, триллионы). Чем больше параметров, тем "умнее" и "значительнее" модель (но не всегда линейно).
- **Температура (Temperature):**
 - Контролирует случайность и креативность ответов (от 0 до 2).
- **Длина контекста (Context Length):**
 - Максимальное количество токенов, которые модель может обработать или "помнить" в одном запросе/ответе (включая промт).
- **Мультимодальность:**
 - Способность модели обрабатывать и генерировать информацию в разных модальностях: текст, изображения, аудио, видео.

Что такое "thinking" / "reasoning"

- LLM не "думают" и не "рассуждают" в человеческом смысле.
- Модели выдают ответы, которые выглядят как результат логического рассуждения, основываясь на предсказании следующего токена.
- Техники: Chain-of-Thought (CoT), Tree-of-Thought, Self-Correction – это методы промптинга, которые заставляют модель "показать" свои промежуточные шаги, улучшая качество рассуждений.

Способы «развертывания»

	«Облака»	Локально
Плюсы	<ul style="list-style-type: none">• Доступны сразу• Не требуется мощное железо• Масштабируемость• Передовые модели• Можно бесплатно	<ul style="list-style-type: none">• Конфиденциальность• Возможность тонкой настройки• *Цена
Минусы	<ul style="list-style-type: none">• Стоимость (за токены)• Вопросы конфиденциальности данных• Зависимость от сторонних сервисов	<ul style="list-style-type: none">• Ограниченный выбор моделей• Не высокая производительность• Сложности с настройкой

«Облачные» модели

Основные «передовые» модели

- Open AI (Chat GPT-4.5)
- Google (Gemini 2.5)
- Anthropic (Claude 4)

Ресурсы для доступа с моделям:

- LM Arena
- Openrouter
- Сайты провайдеров

LM Arena

Looking for something from our old site? Visit legacy.lmarena.ai

LMarena Battle Feedback

New Chat Leaderboard

Today

Привет! Напиши Hellow World ...

Привет! Напиши Hellow World на python.

llama-4-maverick-17b-128e-instruct

Привет!

Вот классический пример "Hello, World!" на Python:

```
print("Hello, World!")
```

stonebloom

Привет! Конечно, вот самая простая и классическая программа "Hello, World!" на Python.

```
print("Hello, World!")
```

Ask followup...

Terms of Use · Privacy Policy · Cookies

+ Chat

Openrouter

The screenshot displays the OpenRouter chat interface. At the top, there is a search bar and a navigation menu. The main chat area shows a conversation with the model 'R1 0528 (free)'. A user message asks for a Python program that outputs 'Hello, world!'. The model's response includes a reasoning phase and a code block with the following Python code:

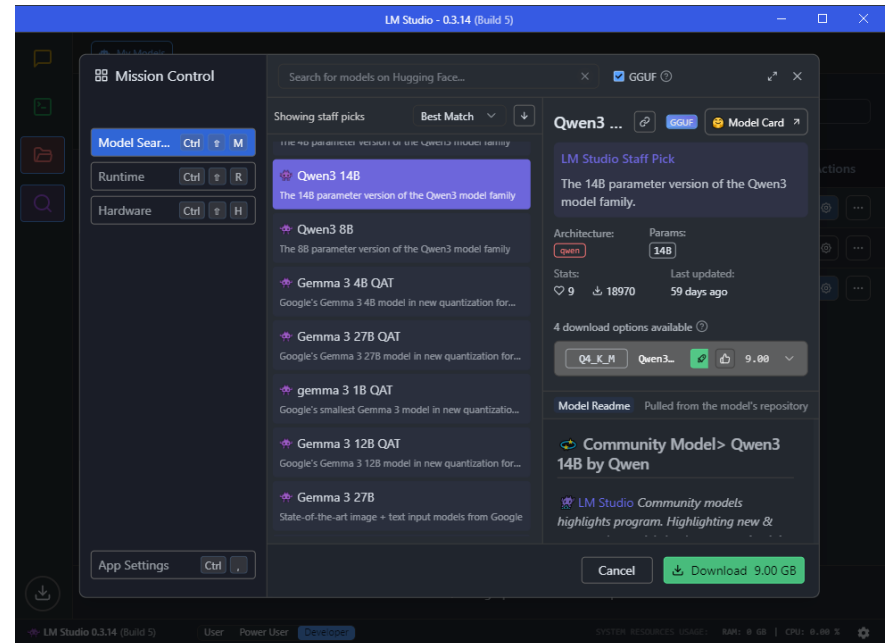
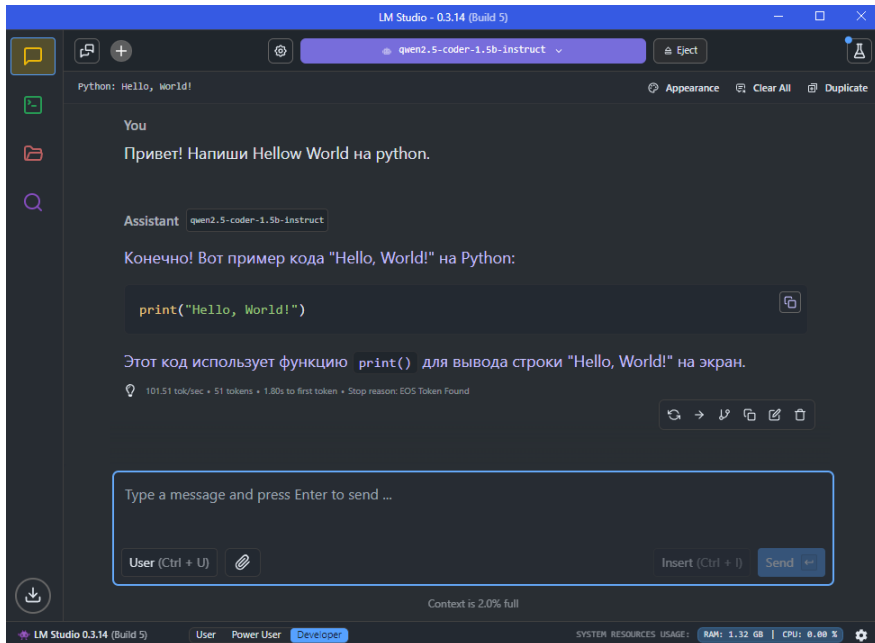
```
python  
  
print("Hello, world!")
```

The interface also features a sidebar with chat history, a bottom input field with a 'Web Search' button, and a 'New Room' button.

Локальные модели







- **Технические требования:**
 - GPU с 8 и более Гб видеопамяти
- **Программы**
 - Библиотеки: llama.cpp или vLLM
 - Консольные: Ollama
 - **С графическим интерфейсом:** LM Studio
- **Модели:**
 - Qwen, llama, gemma, mistral, *deepseek
 - сайт с моделями : <https://huggingface.co>

LM Studio



Как понять, что модель хорошая

- Она находится на высоких позициях в рейтингах
- У нее много параметров
- Она вышла недавно
- Ее хвалят

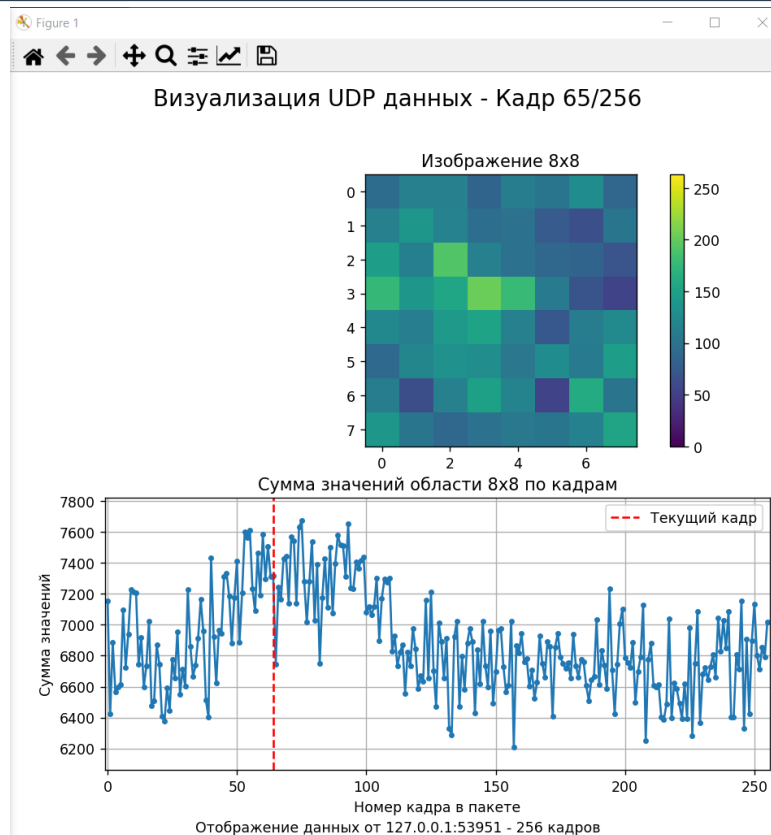
Rank (UB) ↑	Model ↑↓	Score ↑↓	Votes ↑↓
1	 gemini-2.5-pro	1467	12 327
2	 o3-2025-04-16	1451	18 205
2	 gemini-2.5-pro-preview-05-06	1446	14 040
2	 chatgpt-4o-latest-20250326	1442	22 488
3	 gpt-4.5-preview-2025-02-27	1437	15 271
6	 claude-opus-4-20250514	1418	18 287
6	 gemini-2.5-flash	1418	17 535
6	 deepseek-r1-0528	1413	11 871

Правила хорошего промта (Prompt Engineering)

- 1) **Будьте ЧЕТКИМИ и КОНКРЕТНЫМИ:**
 - 1) Что вы хотите получить? (Задача)
 - 2) Какова цель? (Контекст)
 - 3) Какой формат ответа? (Код, JSON, список, таблица)
 - 4) Какие ограничения? (Длина, стиль, тон)
- 2) **Задайте РОЛЬ:** "Выступай в роли [эксперта в области], [опытного программиста], [строгого редактора]..."
- 3) **Предоставьте КОНТЕКСТ:** Объясните предысторию, дайте необходимые данные, ссылки на документы.
- 4) **Используйте ПРИМЕРЫ (Few-shot learning):** Если возможно, покажите модели, как должен выглядеть желаемый вход и выход (например, пара: "плохой код -> хороший код", "неструктурированные данные -> JSON").

Какие задачи решают хорошо?

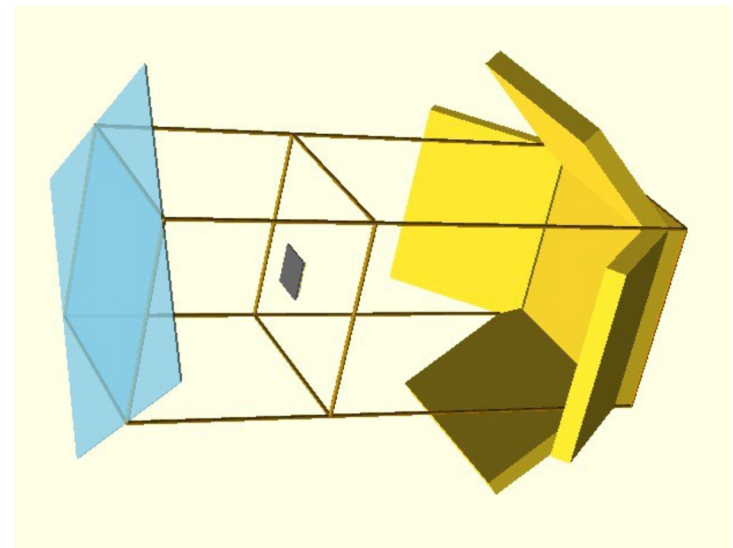
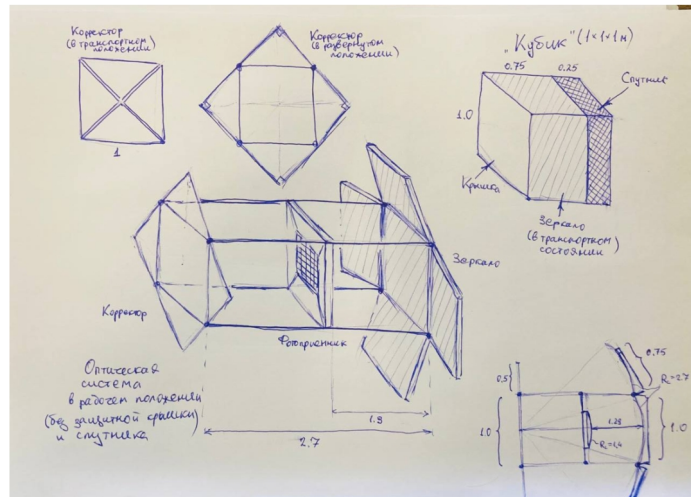
- **Генерация и отладка КОДА:**
 - Простой, структурированный код на популярных языках с использованием общеизвестных библиотек (numpy, pandas, torch).
- **Реферирование и суммаризация:**
 - Научные статьи, технические отчеты, документация, длинные диалоги.
 - Извлечение ключевой информации, фактов, чисел.
- **Генерация текста:**
 - Перефразирование, улучшение стиля.
- **Перевод:**
 - Технических текстов, документации.



Какие задачи решают плохо?

- **Написание кода на редких/узкоспециализированных языках/библиотеках:**
 - Пример: VHDL . Модели обучены на том, что есть в интернете, а там мало такого кода.
- **Сложные, многоэтапные, плохо сформулированные задачи:**
 - Задачи, требующие глубокого доменного экспертного знания, понимания неявных ограничений реального мира, критического мышления и творчества.
- **Задачи, требующие абсолютной точности и отсутствия галлюцинаций:**
 - Например, генерация юридических документов, медицинских диагнозов, критически важного кода для систем жизнеобеспечения.
- **Работа со сложным, нерабочим, "глючным" кодом:**
 - LLM хорошо генерируют новый код. Но попытка разобраться и починить сложный, нерабочий, плохо документированный код с кучей зависимостей, скорее всего, отнимет больше времени, чем написание с нуля или ручная отладка.
- **Задачи, где нет экспертизы у пользователя: Если вы не понимаете предметную область, вы не сможете проверить ответ LLM.**

Необычные задачи



Опасности

- **Галлюцинации:**
 - "Похожесть на правду": LLM генерирует убедительно звучащие, но полностью вымышленные факты, цитаты, ссылки на несуществующие статьи или авторов.
 - Воображение фактов: Особенно опасно в научно-технической деятельности. ВСЕГДА ПРОВЕРЯЙТЕ!
- **"Разучиться писать код / Чересчур полагаться на LLM":**
 - Риск деградации навыков, если полностью делегировать задачи LLM без понимания процесса.
- **Решение задач, где нет экспертизы: Слепое доверие к результатам LLM без возможности их верификации.**
- **Выдача кода за свой**
- **Конфиденциальность данных**

Инструменты

- **AI-IDE (Интеграция в среду разработки):**
 - GitHub Copilot, Codeium, Tabnine: Автодополнение кода, генерация функций, тестов прямо в IDE. Значительно повышает продуктивность.
- **RAG (Retrieval-Augmented Generation - Генерация с дополненным поиском):**
 - Суть: Модель не просто генерирует ответ, но сначала ищет релевантную информацию в предоставленной базе знаний (ваших документах, статьях, базах данных), а затем использует эту информацию для формирования ответа.
- **Finetune (Тонкая настройка):**
- **Агенты (Agents):**
 - Идея: Несколько LLM-агентов взаимодействуют друг с другом, каждый выполняет свою роль (планировщик, исполнитель, критик, редактор), чтобы решить сложную задачу.

Короткое How To: Как начать с нуля

- Попробовать на [lmarena](#)
- Зарегистрироваться на [Openrouter](#), попробовать free модели
- Поставить [LM Studio](#), загрузить [Qwen 3](#)

Практические советы для эффективного использования

- 1) Использовать передовые ("умные") модели: Часто самые мощные модели (GPT-4.5, Claude 4, Gemini 2.5)
- 2) Хорошо структурированное задание с подробным описанием того, что требуется от модели, желательно с примерами. Как бы вы объяснили задачу очень умному, но ничего не знающему о вашем проекте стажеру.
- 3) Один чат - одна задача
- 4) Если задача специализированная, дать дополнительную информацию.
- 5) Всегда проверяйте результат. Рассматривайте LLM как умного, но иногда ошибающегося ассистента, а не как окончательный источник истины.
- 6) Используйте итеративный подход.
- 7) * Не бойтесь экспериментировать с температурой и другими параметрами.
- 8) * Помните о конфиденциальности данных.