



Hugging Face and Open Source Large Language Model



Hugging Face

<https://huggingface.co/>

Компания Hugging Face

Hugging Face — это компания в области искусственного интеллекта, которая разрабатывает и предлагает инструменты машинного обучения и обработки естественного языка (NLP).

Hugging Face служит центром сообщества, где разработчики могут находить модели и наборы данных, делиться ими и использовать.

Основные функции Hugging Face доступны бесплатно.

Существуют также платные тарифные планы. Плата взимается за час (от 0,05 долларов США) или за месяц (от 9 долларов США).

Преимущества Hugging Face

Простота использования. Hugging Face предоставляет удобные инструменты и библиотеки, такие как трансформаторы и наборы данных, которые позволяют разработчикам интегрировать модели с минимальными усилиями.

Сообщество и совместная работа. Участники сообщества участвуют в разработке моделей и инструментов, делятся личным опытом, учебными пособиями и обсуждают последние достижения в области AI.

Широкий выбор моделей. Hugging Face предлагает широкий спектр моделей для решения различных задач искусственного интеллекта от генерации текста до синтеза изображений.

Сравнительный анализ и документация. К каждой модели прилагается подробная документация и контрольные показатели, помогающие выбрать наиболее подходящую модель.

Библиотека 🤗 Transformers

<https://colab.research.google.com/github/huggingface/notebooks/blob/master/course/ru/chapter1/section3.ipynb#scrollTo=0Hi4QuxO6Jk5>

Библиотека Transformers

Transformers — это библиотека современных предварительно обученных моделей с бесплатным доступом.

В настоящее время из библиотеки можно загружать шаблоны для создания текстов, суммаризации статей и извлечения информации (изображений и аудио), а так же шаблоны, комбинирующие эти возможности.

В документацию библиотеки Transformers входят учебные пособия, практические руководства, концептуальные руководства и API.

Существует около 115 моделей с документацией и возможностью интеграции с другими библиотеками, такими как PyTorch, TensorFlow и JAX.

Объект pipeline()

Полный код доступен в Google Colab

<https://colab.research.google.com/github/huggingface/notebooks/blob/master/course/ru/chapter1/section3.ipynb#scrollTo=0Hi4QuxO6Jk5>

Для начала работы необходимо установить Transformers, Datasets и Evaluate:

```
!pip install datasets evaluate transformers[sentencepiece]
```

Объект pipeline()

Базовый объект библиотеки Transformers - pipeline().

Он соединяет в себе необходимые шаги по предобработке и постобработке данных и позволяет передавать модели на вход текст, а на выходе получать читаемый ответ:

```
[ ] from transformers import pipeline

classifier = pipeline("sentiment-analysis")
classifier("I've been waiting for a HuggingFace course my whole life.")
```

```
⇒ [{"label": "POSITIVE", "score": 0.9598047137260437}]
```

Объект pipeline()

Мы можем передать на вход сразу несколько предложений:

```
classifier(  
    ["I've been waiting for a HuggingFace course my whole life.",  
     "I hate this so much!"]  
)
```

```
[{'label': 'POSITIVE', 'score': 0.9598047137260437},  
 {'label': 'NEGATIVE', 'score': 0.9994558095932007}]
```

Объект pipeline()

По умолчанию этот пайплайн выбирает специальную модель, которая была предобучена для оценки тональности предложений на английском языке.

Модель загружается и кэшируется когда создается объект classifier.

Если перезапустить команду, будет использована кэшированная модель, т.е. загрузки новой модели не произойдет.

Некоторые из доступных пайплайнов

- feature-extraction (векторизация текста),
- fill-mask,
- ner (распознавание именованных сущностей),
- question-answering,
- sentiment-analysis,
- summarization,
- text-generation,
- translation,
- zero-shot-classification.

Пайплайн zero-shot-classification

Пайплайн работает с текстами, которые не были размечены (позитивный или негативный, агрессивный или нейтральный и т.д.).

Пайплайн zero-shot-classification позволяет указать, какие метки следует использовать для классификации.

Мы уже видели, как модель может классифицировать предложение как позитивное или негативное. Но модель также может классифицировать текст, используя любой другой набор меток, который подходит для данного корпуса текстов.

Пайплайн `zero-shot-classification`

Этот пайплайн называется *zero-shot* потому, что нет необходимости дообучать модель для использования. Модель возвращает оценки вероятности для любого списка меток.

```
[ ] from transformers import pipeline

classifier = pipeline("zero-shot-classification")
classifier(
    "This is a course about the Transformers library",
    candidate_labels=["education", "politics", "business"],
)
```

```
⇒ {'sequence': 'This is a course about the Transformers library',
   'labels': ['education', 'business', 'politics'],
   'scores': [0.8445963859558105, 0.111976258456707, 0.043427448719739914]}
```

Пайплайн `text-generation`

На вход модели передается небольшой фрагмент текста, который модель будет продолжать.

Генерация текста содержит в себе элемент случайности, поэтому ваш результат может отличаться от того, который приведен в примере.

Можно указать, сколько разных «ответов» сгенерирует модель, задав параметр `num_return_sequences`.

Чтобы изменить длину сгенерированной последовательности, нужно передать значение в аргумент `max_length`.

Пайплайн text-generation

```
[ ] from transformers import pipeline
```

```
generator = pipeline("text-generation")  
generator("In this course, we will teach you how to")
```

```
⇒ [{"generated_text": 'In this course, we will teach you how to understand and use '  
    'data flow and data interchange when handling user data. We '  
    'will be working with one or more of the most commonly used '  
    'data flows – data flows of various types, as seen by the '  
    'HTTP'}]
```

Использование произвольной модели из Hugging Face Hub в пайплайне

Предыдущие примеры использовали модель по умолчанию.

Можно выбрать произвольную модель из Hub и передать ее в пайплайн.

Список доступных моделей можно получить, выполнив следующие действия: перейти по ссылке [Model Hub](https://huggingface.co/models) (<https://huggingface.co/models>), кликнуть на соответствующий тег слева.

Можно выбрать, для какого языка нужна модель, щелкнув на языковые теги, и выбрать ту, которая будет генерировать текст на другом языке.

На Model Hub также есть предобученные модели для многоязычных задач.

Пайплайны: text-generation, использование модели distilgpt2

```
▶ from transformers import pipeline

generator = pipeline("text-generation", model="distilgpt2")
generator(
    "In this course, we will teach you how to",
    max_length=30,
    num_return_sequences=2,
)
```

```
⇒ [{"generated_text": 'In this course, we will teach you how to manipulate the world and '
    'move your mental and physical capabilities to your advantage.'},
    {'generated_text': 'In this course, we will teach you how to become an expert and '
    'practice realtime, and with a hands on experience on both real '
    'time and real'}]
```

Описание модели distilgpt2

DistilGPT2 — англоязычная модель, предварительно обученная с использованием версии GPT-2 с 124 миллионами параметров.

DistilGPT2, который имеет 82 миллиона параметров, был разработан с использованием технологии [knowledge distillation](#) и был задуман как более быстрая и легкая версия GPT-2.

<https://huggingface.co/distilbert/distilgpt2#knowledge-distillation>

Пайплайн fill-mask

Пайплайн решает задачу заполнения пропусков в тексте. Аргумент `top_k` указывает, сколько вариантов для пропущенного слова будет отображено.

```
▶ from transformers import pipeline
```

```
unmasker = pipeline("fill-mask")
```

```
unmasker("This course will teach you all about <mask> models.", top_k=2)
```

```
⇒ [{"sequence": 'This course will teach you all about mathematical models.',  
  'score': 0.19619831442832947,  
  'token': 30412,  
  'token_str': ' mathematical'},  
 {'sequence': 'This course will teach you all about computational models.',  
  'score': 0.04052725434303284,  
  'token': 38163,  
  'token_str': ' computational'}]
```

Пайплайн ner

Пайплайн ner ищет части текста, соответствующие некоторым сущностям, например, персонам, местам, организациям.

```
[ ] from transformers import pipeline
```

```
ner = pipeline("ner", grouped_entities=True)  
ner("My name is Sylvain and I work at Hugging Face in Brooklyn.")
```

```
⇒ [{ 'entity_group': 'PER', 'score': 0.99816, 'word': 'Sylvain', 'start': 11, 'end': 18},  
    { 'entity_group': 'ORG', 'score': 0.97960, 'word': 'Hugging Face', 'start': 33, 'end': 45},  
    { 'entity_group': 'LOC', 'score': 0.99321, 'word': 'Brooklyn', 'start': 49, 'end': 57}  
  ]
```

В этом примере модель корректно обозначила Sylvain как персону (PER), Hugging Face как организацию (ORG) и Brooklyn как локацию (LOC).

Аргумент `grouped_entities=True` передан в пайплайн для того, чтобы модель сгруппировала части предложения, соответствующие одной сущности: в данном случае модель объединила “Hugging” и “Face”.

Пайплайн question-answering

Пайплайн позволяет сгенерировать ответ на вопрос по данному контексту.

```
[ ] from transformers import pipeline

question_answerer = pipeline("question-answering")
question_answerer(
    question="Where do I work?",
    context="My name is Sylvain and I work at Hugging Face in Brooklyn",
)
```

Пайплайн question-answering

Пайплайн позволяет сгенерировать ответ на вопрос по данному контексту.

```
[ ] from transformers import pipeline

question_answerer = pipeline("question-answering")
question_answerer(
    question="Where do I work?",
    context="My name is Sylvain and I work at Hugging Face in Brooklyn",
)
```

Пайплайн summarization

```
▶ from transformers import pipeline

summarizer = pipeline("summarization")
summarizer(
    """
    America has changed dramatically during recent years. Not only has the number of
    graduates in traditional engineering disciplines such as mechanical, civil,
    electrical, chemical, and aeronautical engineering declined, but in most of
    the premier American universities engineering curricula now concentrate on
    and encourage largely the study of engineering science. As a result, there
    are declining offerings in engineering subjects dealing with infrastructure,
    the environment, and related issues, and greater concentration on high
    technology subjects, largely supporting increasingly complex scientific
    developments. While the latter is important, it should not be at the expense
    of more traditional engineering.

    Rapidly developing economies such as China and India, as well as other
    industrial countries in Europe and Asia, continue to encourage and advance
    the teaching of engineering. Both China and India, respectively, graduate
    six and eight times as many traditional engineers as does the United States.
    Other industrial countries at minimum maintain their output, while America
    suffers an increasingly serious decline in the number of engineering graduates
    and a lack of well-educated engineers.
    """
)
```

Пайплайн
решает
задачу
реферирования
текста.

Пайплайн summarization

```
⇒ [ {'summary_text': ' America has changed dramatically during recent years . The '
    'number of engineering graduates in the U.S. has declined in '
    'traditional engineering disciplines such as mechanical, civil '
    ', electrical, chemical, and aeronautical engineering . Rapidly '
    'developing economies such as China and India, as well as other '
    'industrial countries in Europe and Asia, continue to encourage '
    'and advance engineering .' } ]
```

Так же, как и в задаче генерации текста, можно указать максимальную длину `max_length` или минимальную длину `min_length` результата, например:

```
summary = summarizer(text,
                      max_length=130,
                      min_length=30,
                      do_sample=False)
```

Пайплайн translation

Пайплайн выполняет перевод с одного языка на другой.

```
[ ] from transformers import pipeline

    translator = pipeline("translation", model="Helsinki-NLP/opus-mt-fr-en")
    translator("Ce cours est produit par Hugging Face.")
```

```
⇒ [ {'translation_text': 'This course is produced by Hugging Face.'} ]
```

Так же, как и в задачах генерации и автоматического реферирования текста, вы можете указать максимальную длину `max_length` или минимальную длину `min_length` результата.

Библиотека Transformers

Модели трансформеров обычно очень большие. С миллионами и десятками *миллиардов* параметров, обучение и развертывание этих моделей — сложная задача.

Кроме того, регулярно выходят новые модели и каждая из них имеет свою собственную реализацию.

Для решения этой проблемы была создана библиотека  Transformers.

Ее цель — предоставить единый API, с помощью которого можно загрузить, обучить и сохранить любую модель Transformer.

Особенности библиотеки 🤖 Transformers

Простота использования. Скачать, загрузить и использовать современную модель NLP для инференса можно всего в две строчки кода.

Гибкость. Все модели представляют собой простые классы `PyTorch nn.Module` или `TensorFlow tf.keras.Model` и могут быть обработаны как любые другие модели в соответствующих фреймворках машинного обучения (ML).

Простота. В библиотеке почти нет абстракций. Концепция «Все в одном файле» является основной: прямой проход модели полностью определяется в одном файле, так что сам код понятен и доступен для изменения.

Эта особенность отличает 🤖 Transformers от других ML-библиотек. Модели не строятся на основе модулей которые совместно используются в разных файлах, вместо этого каждая модель имеет свои собственные слои.

Курс от Hugging Face

<https://huggingface.co/learn/nlp-course/ru/chapter0/1>

Open Source LLM

Large Language Model

Зачем запускать LLM локально

Преимущества Open Source моделей по сравнению с использованием ChatGPT, Google Gemini, Claude 3 и тд:

Бесплатно. Вам не нужно платить за подписку. И если \$20 в месяц для личного использования — не критично, то при разработке своего проекта можно хорошо сэкономить на API.

Оффлайн доступ. Модель запускается локально — вам не нужно подключение к интернету, VPN и тд.

Приватно. Ваши данные никуда не отправляются, все хранится локально на компьютере. Можно не бояться, что корпорации получат доступ к личной информации.

Персонализация. Используя такие подходы, как RAG (Retrieval Augmented Generation), вы можете обогатить модели большим объемом своих данных — бесплатно дать доступ к тысячам документов, хранящихся на вашем компьютере.

LLM. Минусы

1. Требования к железу. Хорошую LLM не получится запустить на слабом компьютере. Но, например, если у вас есть MacBook с чипом M1 и выше и хотя бы 16GB ОЗУ, то вам уже доступно множество моделей, качество которых будет сравнимо с ChatGPT 3.5 Turbo. Правда, это касается в основном запросов на английском языке, с русским языком Open Source работает хуже.
2. Нет синхронизации между девайсами. Так как все происходит локально, то продолжить диалог с моделью на телефоне не получится.

Локальный запуск. LM Studio

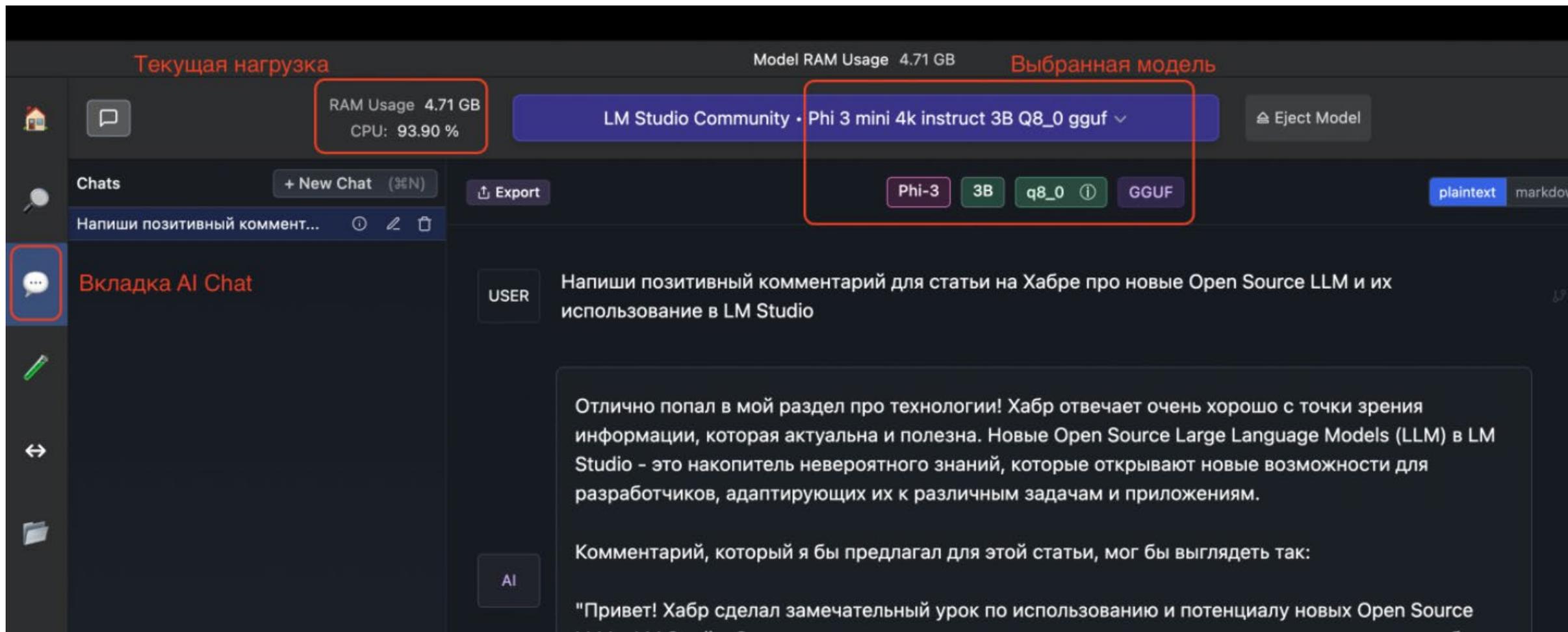
LM Studio предоставляет удобный графический интерфейс для загрузки всех популярных LLM и общения с ними.

Чтобы начать работу, вам нужно:

- 1.Скачать и установить LM Studio с [официального сайта](#)
- 2.В интерфейсе кликнуть "Download" в карточке нужной модели
- 3.Перейти во вкладку "AI Chat" и писать запросы в выбранную Open Source LLM

Вкладка AI Chat выглядит по сути так же, как веб-интерфейс ChatGPT: слева список диалогов с моделью, по центру чат.

Локальный запуск. LM Studio



Локальный запуск. Ollama

Ollama — это приложение для Mac, Linux и Windows, предоставляющее быстрый доступ к Open Source LLM через CLI интерфейс.

Здесь, в отличие от LM Studio, нет графического интерфейса, то есть писать запросы нужно в терминале. Но разработчики утверждают, что так даже удобнее, ведь все взаимодействие с LLM происходит через текст, а GUI только тормозит работу.

Локальный запуск. Ollama

Ollama — это приложение для Mac, Linux и Windows, дающее быстрый доступ к Open Source LLM через CLI интерфейс.

Здесь, в отличие от LM Studio, нет графического интерфейса, то есть писать запросы нужно в терминале. Но разработчики утверждают, что так даже удобнее, ведь все взаимодействие с LLM происходит через текст, а GUI только тормозит работу.

Для начала работы вам потребуется:

1. Установить Ollama с [официального сайта](#).
2. Написать в консоли команду `ollama run llama3`. Вместо llama3 можно выбрать любую другую Open Source LLM.
3. Писать вопросы к модели прямо в терминале.

Локальный запуск. Ollama

```
nikitayusupov@Nikitas-MacBook-Pro ~ % ollama run llama3
>>> Привет! Напиши позитивный комментарий для Habr про статью об использовании Ollama
"Восхитительный обзор использования Ollama! Я никогда не думал о том
инструмент может изменить мой способ работы с данными. Теперь я пони
почему он так популярен в сообществе разработчиков. Описание шагов, которые в
нужно выполнить для настройки и использования Ollama, сделано очень
доступно. Я особенно рад, что автор поделился собственным опытом и советом
том, как использовать это инструмент в повседневной работе. Спасибо з
полезную статью! "
```

```
>>> █ end a message (/? for help)
```

Локальный запуск. Ollama

Требования к железу, как обычно, зависят от конкретной LLM, но ваша GPU должна иметь Compute Capability 5.0 или выше. Список таких видеокарт можно посмотреть [здесь](#).

Качество работы на русском языке оставляет желать лучшего, лучше писать запросы на английском.

Код на Python

Конечно, веса любой модели можно просто скачать с Hugging Face и запустить через Python, но если эта модель уже доступна в описанных выше редакторах, то обычно проще воспользоваться готовым GUI или CLI интерфейсом.

Новые Open Source LLM

Phi-3-vision

Первая мультимодальная LLM в семействе Phi-3 от Microsoft.

Модель понимает не только текст, но и фото.

Попробовать бесплатно эту модель можно:

- на ее [странице](#) в Azure AI Studio,
- запустить локально, используя код с [Hugging Face](#) (хватает 12GB на видеокарте).

Phi-3-vision. Пример использования

Отправляем модели скриншот таблицы и попросить модель переписать ее в виде Markdown.

Benchmark	Phi-3 Vision- 128K-In1	LlaVA-1.6 Vicuna-7B	QWEN- VL Chat
MMMU	40.2	34.2	39.0
MMBench	80.5	76.3	75.8
ScienceQA	90.8	70.6	67.2
MathVista	44.5	31.5	29.4

Extract the table content of the table in Markdown format:

Phi-3-vision. Пример использования

Certainly! Below is the markdown table converted from the image:

Benchmark	Phi-3 Vision-128K-In1	LLaVA-1.6 Vicuna-7B	QWEN-VL Chat
MMMU	40.2	34.2	39.0
MMBench	80.5	76.3	75.8
ScienceQA	90.8	70.6	67.2
MathVista	44.5	31.5	29.4

This table compares different benchmarks across three different AI models: Phi-3 Vision-128K-In1, LLaVA-1.6 Vicuna-7B, and QWEN-VL Chat. The benchmarks include MMMU, MMBench, ScienceQA, and MathVista, with their respective scores for each AI model.

Phi-3-vision. Пример использования

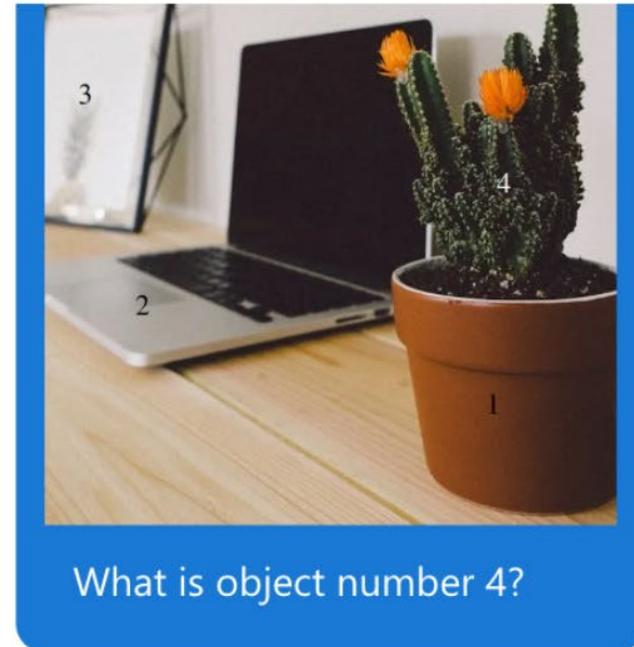
Как мы видим, модель отлично понимает структуру и смысл картинки.

Аналогично вы можете описывать разные графики, диаграммы, и просто любой объект на фото из окружающего мира.

Еще один пример, где модель распознала конкретный объект на фото, помеченный цифрой.

Phi-3-vision. Пример использования

Попробовать ⓘ



Object number 4 is a cactus in a pot.



Начните вводить здесь



Mistral-7B v0.3

Это обновление предыдущей версии Mistral-7B-v0.2, которая занимала 52 место в рейтинге [Chatbot Arena](#), что выше, чем у GPT-3.5.

В версии 0.3 появилась поддержка Function Calling — это значит, что вы можете дать модели информацию о некоторых функциях, которая она будет предлагать вызывать в зависимости от контекста.

Например, вы можете описать функцию `get_weather(city)`, и если модель получит вопрос о текущей погоде в каком-то городе, она предложит вызвать эту функцию и передать в аргументы нужный город. Это полезно, если вы разрабатываете свой проект на базе LLM и хотите расширить стандартный функционал модели.

Новую версию можно попробовать прямо в Web интерфейсе на Hugging Face, также она доступна через Ollama.

CogVLM2

Это мультимодальная модель, основанная на [Llama-3-8B-Instruct](#). Авторы утверждают, что она превосходит (или хотя бы не хуже) GPT-4 Vision.

Модель имеет 19B параметров, что больше, чем у Phi-3 Vision.

Для запуска этой модели потребуется от 16GB видео-памяти.

Модель может анализировать картинки разрешением до 1344 * 1344 пикселей, а также поддерживать диалог с длиной контекста в 8K токенов.

Работает для английского и китайского языков.

Скачать модель, чтобы запустить локально, можно на [Hugging Face](#).

Ллама-3 8B

Модель занимает 19 место в рейтинге на [Chatbot Arena](#), обходя все версии ChatGPT-3.5 Turbo и многие другие.

Ллама 3 можно запустить через [Ollama](#) или скачать на [официальном сайте](#).

Благодаря 4-битной квантизации, модель занимает всего 4.7 GB.

Chatbot Arena

<https://lmarena.ai/>

Chatbot Arena

Chatbot Arena — это система, которая позволяет тестировать и сравнивать различные языковые модели нейросетей, оценивать их производительность, а также настраивать параметры тестирования в соответствии с требованиями проекта и выбирать наиболее эффективный вариант.

Платформа основана на рейтинговой системе Elo, заимствованной из шахматного мира. Она выступает надёжным механизмом для сравнения — по этому принципу можно оценивать практически неограниченное количество парных сочетаний нейросетей. Во время тестирования языковых моделей сервис собирает сведения о возможностях применения каждой нейросети для различных задач.

Как проходит сравнение моделей

На ChatBot Arena собрано множество языковых моделей для сравнения друг с другом, включая такие крупные, как GPT-4 от OpenAI и [Claude](#) от Anthropic. Также здесь представлены старые версии GPT и другие нейросети с открытым доступом.

[Официальный сайт](#) сервиса предлагает несколько вариантов тестирования и сравнения моделей. В режиме «битвы» (Battle) названия нейросетей не отображаются, вы проверяете ответы на запрос от двух систем одновременно, при этом не зная, какая из них срabатывает в данный момент. В форме открытого сравнения (Side-by-Side) вы сами можете выбирать из списка, какие модели хотите протестировать.

Как проходит сравнение моделей

Для полноценного теста нужно задать несколько вопросов в поле ввода, пока не станет ясно, какой чат-бот из двух отвечает лучше. Когда вынесете свой вердикт, нажмите на одну из кнопок, которая подтвердит ваше решение: «А лучше» или «В лучше». Также можно выбрать вариант «Ничья», если оба чат-бота показали одинаково хорошие результаты, или «Оба плохие», если ни один из их ответов не понравился.

После того как вы определите победителя, в режиме битвы ChatBot Arena автоматически попросит каждого бота подтвердить свою «личность», чтобы вы поняли, какая модель в лидерах.

Результаты обычно зависят от того, какие запросы вы даёте.

Метод RAG (Retrieval Augmented Generation)

Поисковая расширенная генерация

<https://python.langchain.com/docs/tutorials/rag/>

Что такое RAG

RAG (Retrieval Augmented Generation) — это метод работы с большими языковыми моделями, когда пользователь пишет свой вопрос, а вы **программно** к этому вопросу «подмешиваете» дополнительную информацию из каких-то внешних источников и подаете все целиком на вход языковой модели.

Другими словами вы добавляете в контекст запроса к языковой модели дополнительную информацию, на основе которой языковая модель может дать пользователю более полный и точный ответ.

Что такое RAG. Пример 1

- Пользователь спрашивает модель: «какой сейчас курс доллара?»
- Очевидно, что языковая модель понятия не имеет какой СЕЙЧАС курс доллара, эту информацию она должна откуда-то получить, чтобы ответить на вопрос.
- Что нужно сделать? Нужно открыть первую ссылку в Google по запросу «курс доллара к рублю» и содержимое страницы добавить к вопросу пользователя, чтобы LLM могла правильно ответить на вопрос используя эту информацию.

Что такое RAG. Пример 2

- Допустим вы делаете автоматического оператора технической поддержки на основе LLM. У вас есть база знаний вопросов и ответов или просто подробное описание функциональности.
- Самое плохое, что можно сделать, это дообучить модель на вашу базу знаний вопрос-ответ. Почему это плохо? Потому что база знаний будет постоянно меняться и постоянно дообучать модель это дорого и не эффективно.
- Нужно использовать RAG. В зависимости от того, какой вопрос задал пользователь ваша RAG система должна найти соответствующую статью в базе знаний, и подать на вход LLM не только вопрос пользователя, но и релевантную запросу часть содержимого базы знаний, чтобы LLM могла сформировать правильный ответ.

Что такое RAG

Таким образом фраза Retrieval Augmented Generation как нельзя более точно описывает суть происходящего:

Retrieval - поиск и извлечение релевантной информации. Часть системы, которая отвечает за поиск и извлечение информации, так и называют — ретривер (retriever).

Retrieval Augmented — дополнение запроса пользователя найденной релевантной информацией.

Retrieval Augmented Generation — генерация ответа пользователю с учетом дополнительно найденной релевантной информации.

RAG. Трудности

1. Нечеткий поиск — просто взять запрос пользователя и найти по точному соответствию все куски из базы знаний не получится. Каким алгоритмом реализовать поиск, чтобы он искал релевантные и только релевантные «куски» текста?
2. Размер статей из базы знаний — какого размера «куски» текста надо давать LLM, чтобы она по ним формировала ответ?
3. А если в базе знаний нашлось несколько статей? А если они большие? Как их «обрезать», как комбинировать, может быть сжать?

RAG. Базовый алгоритм

База знаний «нарезается» на небольшие куски текста, так называемые chunks (чанки). Размер чанков может быть от нескольких строк до нескольких абзацев (примерно от 100 до 1000 слов). Затем чанки оцифровываются с помощью «эмбеддера» и превращаются в эмбеддинги (вектора).

Когда пользователь отправляет свой вопрос в LLM, то текст его запроса точно по такому же алгоритму (как правило, тем же самым эмбеддером), кодируется также в еще один эмбеддинг и далее над базой данных содержащей наши эмбеддинги чанков производится поиск наиболее близких «по смыслу» эмбеддингов (векторов).

В реальности, как правило, считается косинусная близость вектора запроса и вектора каждого чанка и далее выбираются топ N векторов наиболее близких к запросу.

Далее текст чанков, соответствующий этим найденным векторам, вместе с запросом пользователя объединяется в единый контекст и подается на вход языковой модели. т. е. модель «думает», что пользователь написал ей не только вопрос, но еще и предоставил данные на основе которых нужно ответить на поставленный вопрос.

RAG. Базовый алгоритм

Как видно из описания ничего сложного: разбили текст на куски (чанки), каждый кусок оцифровали, нашли наиболее близкие по смыслу куски и подали текст этих кусков на вход большой языковой модели вместе с запросом от пользователя. Более того, весь этот процесс как правило уже реализован в так называемом пайплайне и все, что вам нужно, это собственно запустить пайплайн (из какой-нибудь готовой библиотеки).

Написав первую версию и подав на вход несколько документов можно убедиться в том, что полученный алгоритм в 100% случаев работает не так, как от него ожидалось и далее начинается долгий и сложный процесс настройки RAG.

Настройка RAG. Размер чанков и их количество

Имеет смысл начать эксперименты с изменения размера чанков и их количества. Если вы подаете на вход LLM слишком много ненужной информации или наоборот, слишком мало нужной информации, то вы просто не оставляете шансов модели ответить правильно:

- Чем меньше чанк по размеру, тем точнее будет буквальный поиск, чем больше размер чанка тем больше поиск приближается к смысловому.
- Разные запросы пользователя могут содержать разное количество чанков, которое необходимо добавлять в контекст. Необходимо опытным путем подобрать тот самый коэффициент, ниже которого чанк смысла не имеет и будет лишь замусоривать ваш контекст.
- Чанки должны перекрывать друг друга, чтобы у вас был шанс подать на вход последовательность чанков, которые следуют друг за другом вместе, а не просто вырванные из контекста куски.
- Начало и конец чанка должны быть осмысленными, в идеале должны совпадать с началом и концом предложения, а лучше абзаца.

Настройка RAG. Другие методы поиска

Часто поиск «по смыслу» через эмбединги не дает нужного результата. Особенно если речь идет о каких-то специфических терминах или определениях.

Как правило к поиску через эмбединги подключают также TF-IDF поиск и объединяют результаты поиска в пропорции, подобранной экспериментальным путем.

Также очень часто помогает ранжирование найденных результатов, например, алгоритмом BM25.

Настройка RAG. Мультиплицирование запроса и суммаризация чанков

Мультиплицирование запроса. Как правило, запрос от пользователя имеет смысл несколько раз перефразировать (с помощью LLM) и осуществлять поиск чанков по всем вариантам запроса. На практике делают от 3-х до 5-ти вариаций запросов и потом результаты поиска объединяют в один.

Суммаризация чанков. Если по запросу пользователя найдено очень много информации и вся эта информация не помещается в контекст, то ее можно точно также «упростить» с помощью LLM и подать на вход в виде контекста (в добавление к вопросу пользователя) нечто вроде заархивированного знания, чтобы LLM могла использовать суть (выжимку из базы знаний) при формировании ответа.

Настройка RAG. Системный промпт и дообучение модели на формат RAG

Чтобы модель лучше понимала, что от нее требуется возможно также дообучить LLM на правильный формат взаимодействия с ней.

В подходе RAG контекст всегда состоит из двух частей (мы пока не рассматриваем вариант диалога в формате RAG): вопроса пользователя и найденного контекста.

Соответственно модель можно дообучить понимать именно вот такой формат: тут вопрос, тут информация для ответа, выдай ответ.

На первоначальном этапе эту проблему можно также попытаться решить через системный промпт, объяснив модели, что «вопрос тут, информация тут».

Настройка RAG. Оценка качества работы

Вопросы для проверки — должны быть написаны человеком.

Только вы (или ваш заказчик) знает какие вопросы будут задаваться системе и ни кто, кроме вас проверочные вопросы не напишет.

Референсные ответы — тоже в идеале должны быть написаны людьми, и желательно разными и в двух (а то и трех) экземплярах, чтобы избавиться от зависимости от человеческого фактора. Но тут возможны другие варианты. Например Ассистенты от OpenAI на модели GPT Turbo достаточно хорошо справляются с этой задачей, особенно если в них грузить не слишком большие (не более 100 страниц) документы. Для получения референсных ответов, можно написать скрипт, который обратится к ассистенту (в который загружены релевантные документы) и задаст ему все ваши вопросы для проверки. Причем несколько раз один и тот же вопрос.

Настройка RAG. Оценка качества работы

Близость ответов LLM к референсным — измеряется несколькими метриками, такими как BERTScore, BLEURT, METEOR и даже простым ROUGE. Как правило опытным путем подбирается средневзвешенная метрика, являющаяся суммой вышеперечисленных (с соответствующими коэффициентами) которая наиболее точно отражает реальную близость ответов вашей LLM к правильным ответам.

Референсные чанки и близость найденных чанков к референсным. Чанки, которые подаются на вход модели, обеспечивают 80% качества ответа. Поэтому необходимо создать базу данных из вопросов и соответствующим им чанков и каждый раз проверять, насколько точно ретривер нашел чанки, соответствующие запросу от пользователя.