

Применение LLM

Промптинг – как найти общий язык с моделью

GIGA^{2.0}
CHAT

Ты - преподаватель машинного обучения и эксперт в LLM. Расскажи что такое промптинг

Промптинг (от англ. prompting) – это практика составления вводных инструкций или подсказок для большой языковой модели (Large Language Model, LLM), направляющих её поведение и улучшающих качество генерируемых ответов. Промпт служит отправной точкой для моделирования реакции на заданный запрос, определяя контекст, тон, стиль и структуру желаемого результата.

Простым языком, промпт – это начальная фраза или набор инструкций, которые задают направление модели для правильного восприятия и интерпретации входящего запроса. Правильно составленные промпты помогают извлекать максимальную пользу из возможностей LLM, добиваясь более точных, релевантных и полезных результатов.

Промптинг – как найти общий язык с моделью

Зачем нужен промптинг?

1. **Уточнение цели:** Помогает четче сформулировать задачу перед моделью, предотвращая двусмысленность или недопонимание.
2. **Контроль качества:** Позволяет влиять на стиль, форму, длину и сложность ответа.
3. **Повышение точности:** Уменьшает вероятность ошибок или неправильных ответов, создавая ясные инструкции для конкретной задачи.
4. **Эффект демонстрации:** Показывая модели желаемый формат ответа (например, через примеры), мы получаем более предсказуемые и качественные результаты.

Типичные виды промптинга:

1. **Zero-shot prompting:** Когда задание даётся без примеров, полагаясь исключительно на общие знания модели.
2. **One-shot prompting:** Включает один показательный пример вместе с задачей.
3. **Few-shot prompting:** Предоставляет несколько примеров, демонстрирующих правильное исполнение задания.
4. **Chain-of-thought prompting:** Модели предлагается пошагово описать своё мышление, помогая ей разобраться в сложной задаче.
5. **Instruction prompting:** Четко формулируется ↓ инструкция с описанием задачи и требований к ответу.

Zero shot

Переведи на английский: 'Чужая душа потёмки.'. Используй стиль поговорок

"Another person's heart is a mystery."

Или коротко и ёмко:

"You can't see into someone else's mind."



Few shot

Переведи на английский: Пример 1: "льет как из ведра" - "rains cats and dogs" "Чужая душа потёмки" - ?

«Чужая душа потёмки» — «You can't tell what's going on inside someone else».

Однако наиболее близкий аналог русской пословицы среди английских идиом — это выражение: — "**The heart has its reasons that reason knows not of**" (Сердце имеет причины, неизвестные разуму), хотя оно звучит чуть сложнее и несет другой оттенок смысла.

Чаще всего используют простое и понятное выражение: — "**Other people's minds are hard to fathom**" (чужие мысли трудно постичь).

Chain of thoughts

Chain-of-Thought Prompting Elicits Reasoning in Large Language Models
<https://arxiv.org/abs/2201.11903>

GIGACHAT 2.0 ⓘ

Ты - эксперт в области LLM из OpenAI. Расскажи как работает chain of thoughts

Chain-of-Thought Prompting (промтинг методом цепочки рассуждений) — это методика работы с большими языковыми моделями (LLM), которая основана на постепенном развертывании процесса принятия решений, разделяя сложное задание на шаги и промежуточные выводы. Этот подход направлен на улучшение способности модели рассуждать, систематически подходить к решению задач и уменьшать число случайных ошибок.

Ключевые идеи Chain-of-Thought

- 1. Разделение задачи на этапы.** Вместо прямого ответа на вопрос, модель получает инструкцию постепенно двигаться шаг за шагом, рассматривая отдельные части задачи и приходя к финальному выводу.
- 2. Формулировка промежуточных рассуждений.** Модель должна сначала объяснить свою мысль, аргументировать выбор каждого шага и только после этого переходить к следующему этапу. Таким образом, увеличивается прозрачность и логика выводов.
- 3. Использование вспомогательных аргументов.** Часто используются различные правила, формулы или известные факты, помогающие модели прийти к верному решению.

Chain of thoughts

GIGACHAT 2.0 ⓘ

Пример работы Chain-of-Thought

Допустим, задача такая: "Петя купил 7 яблок, съел 3 яблока и подарил 2 своим друзьям. Сколько яблок осталось?"

Вот как выглядит Chain-of-Thought reasoning:

1. Петя изначально имел 7 яблок.
2. Затем он съел 3 яблока: $7 - 3 = 4$ яблока остались.
3. После этого он отдал ещё 2 яблока друзьям: $4 - 2 = 2$ яблока остались.
4. Финальный вывод: у Пети осталось 2 яблока.

Каждый шаг здесь важен, так как показывает путь к конечному ответу и снижает риск ошибок, возникающих при попытках немедленно вычислить итоговое значение.

RAG

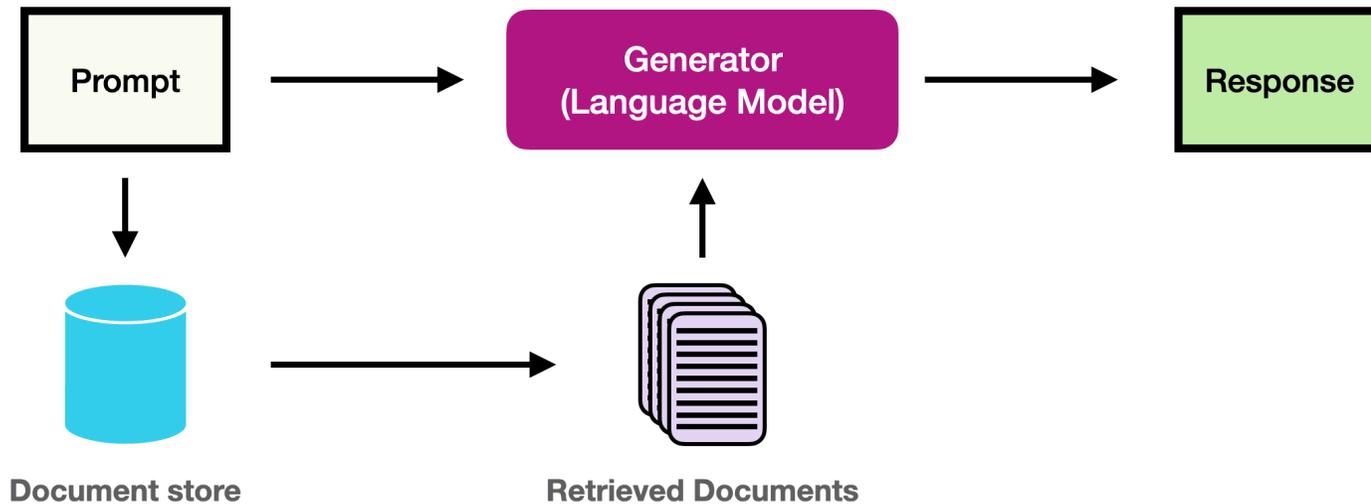
Проблемы предобученных LLM:

- Обучены кем-то на каких-то данных
- Устаевают в момент завершения обучения
- Галлюцинируют – выдают ложную, выдуманную информацию за действительную
- Могут не знать специфики и предметной области вашей задачи

RAG (Retrieval-Augmented Generation) — это метод улучшения производительности больших языковых моделей путем интеграции внешних источников информации, таких как базы данных, веб-сайты или другие хранилища данных, непосредственно во время процесса генерации текста. Основная идея заключается в том, чтобы обеспечить модели доступ к актуальной и точной информации, повышая таким образом качество и точность её ответов.

RAG

Retrieval Augmented Generation



- 1) В хранилище знаний помещаются документы, по которым нужно вести поиск.
- 2) Запрос пользователя передается в хранилище знаний, где ищутся наиболее схожие документы.
Retreive-часть
- 3) На вход LLM передается и сам запрос, и найденные документы.
Augmentation-часть
- 4) В промпте к LLM указывается, что отвечать нужно строго в соответствии с представленными документами.
Generation-часть

Успех RAG на 90% зависит от качества Retreive-части и полноты базы знаний

RAG – легко на словах...

1) Хранилище данных – векторное хранилище, где хранятся эмбединги кусков документов.

На какие куски разбивать? (например, на $500 \pm$ токенов) А разбивать ли вообще?

Какую модель использовать для получения эмбедингов?

Чувствительность к сравнению текстов разной длины

2) Для работы с разномодальными данными (текст, изображения, аудио,...) требуются разные агенты Retrieve-части, т.к. поиск по таким данным скорее всего должен быть разным.

3) Запрос пользователя может требовать дополнительной обработки. В том числе, с применением LLM.

4) Результаты retrieve-части могут дополнительно обрабатываться с помощью LLM, например – суммаризироваться.

5) Требуются хитрые промпты для запроса Generate-части

6) Как оценить итоговое качество системы?

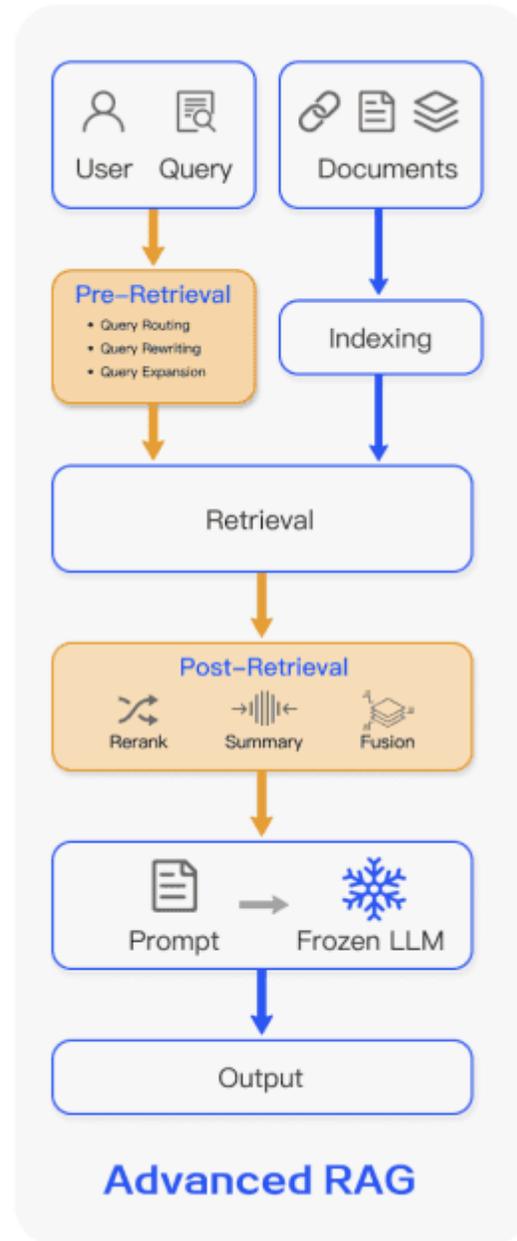
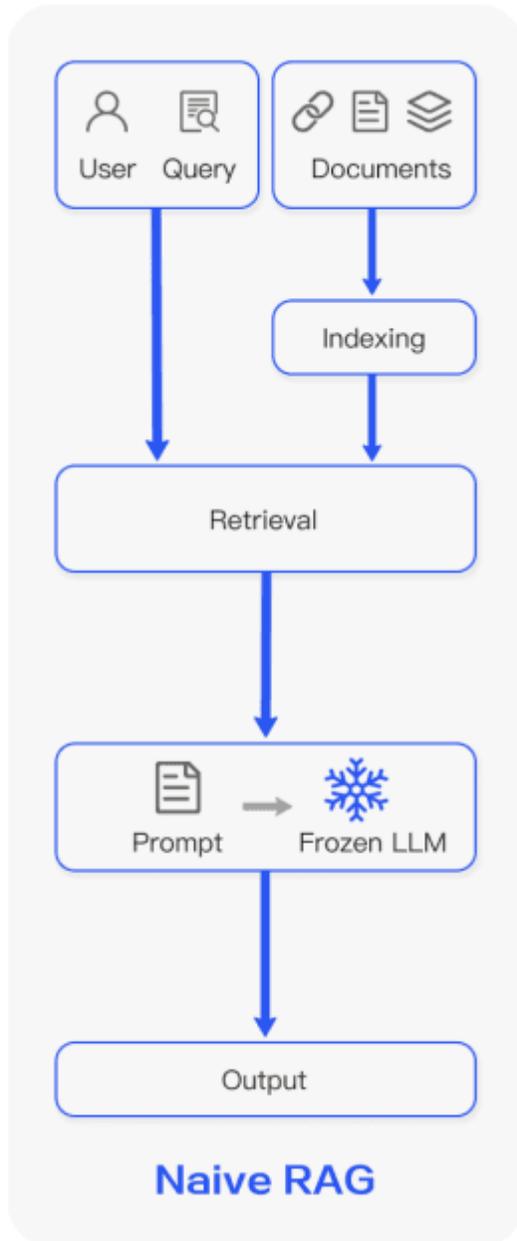
Сравнивать схожести эмбединга текста, сгенерированного системой с эмбедингом правильного ответа эксперта.

LLM as a judge

RAG – легко на словах...

```
prompt = "Вы – эксперт по работе с базой знаний Wikipedia. Анализируя приведенные документы:\n"  
prompt+= "1. Отвечайте ТОЛЬКО на основе предоставленных данных\n"  
prompt+= "2. Для каждого факта указывайте номер документа в квадратных скобках\n"  
prompt+= "3. При отсутствии информации явно сообщите об этом\n"  
prompt+= "4. Форматируйте ответ как:\n"  
prompt+= "    - Краткий ответ\n"  
prompt+= "    - Детализация цитатами\n"  
prompt+= "    - Список использованных документов с их названиями\n\n"  
prompt+= f"Вопрос: {query}\n\n"  
prompt+= "Доступные документы:\n"  
  
# Добавляем тексты с номерами и источниками  
for i, (text, doc) in enumerate(zip(relevant_texts, relevant_docs), 1):  
    source = doc.get('records', 'Источник не указан')  
    prompt += f"[{i}] {text.strip()} (Источник: {doc})\n"  
  
prompt += "Требования:\n- Ответы должны генерироваться только на русском языке\n- Максимальная фактологическая точность\n ' \n- Разделение собственных выводов и цитат\n- Названия документов должны точно соответствовать источникам\n\n\n":
```

RAG – легко на словах...



Существуют готовые фреймворки, разворачивающие RAG, например PrivatGPT
<https://docs.privategpt.dev/overview/welcome/introduction>

Но подходят только для простых задач.

RAG – джентельменский набор



фреймворк с открытым исходным кодом для разработки приложений на основе больших языковых моделей (LLM)



фреймворк, который позволяет загружать, запускать и управлять большими языковыми моделями (LLM) локально, на компьютере



Векторные базы данных.



Weaviate

Оптимальное хранение, управление и поиск векторов-эмбеддингов