

# Prompt engineering

Это руководство делится стратегиями и тактиками для получения лучших результатов от больших языковых моделей (иногда их называют моделями GPT), таких как GPT-4. Описанные здесь методы иногда можно использовать в комбинации для большего эффекта. Мы рекомендуем экспериментировать, чтобы найти методы, которые лучше всего работают для вас.

Некоторые примеры, демонстрируемые здесь, в настоящее время работают только с нашей наиболее мощной моделью `gpt-4`. В общем, если вы обнаружите, что модель не справляется с задачей и доступна более мощная модель, часто стоит попробовать снова с более мощной моделью.

## Шесть стратегий для получения лучших результатов

### Пишите четкие инструкции

Эти модели не могут читать ваши мысли. Если результаты слишком длинные, попросите дать краткие ответы. Если результаты слишком простые, попросите написать текст уровня эксперта. Если вам не нравится формат, продемонстрируйте желаемый формат. Чем меньше модели приходится догадываться о вашем желании, тем больше шансов, что вы его получите.

Тактики:

- Включите детали в ваш запрос, чтобы получить более релевантные ответы
- Попросите модель принять определенную роль
- Используйте разделители, чтобы четко указать различные части ввода
- Укажите шаги, необходимые для выполнения задачи
- Предоставьте примеры
- Укажите желаемую длину вывода

### Предоставьте справочный текст

Языковые модели могут с уверенностью изобретать недостоверные ответы, особенно когда их спрашивают о эзотерических темах или просят цитаты и URL-адреса. Так же, как листок с заметками может помочь студенту лучше сдать тест, предоставление справочного текста этим моделям может помочь в ответах с меньшим количеством вымысла.

Тактики:

- Инструктируйте модель отвечать, используя текстовый источник
- Инструктируйте модель отвечать с цитатами из текстового источника

### **Разбивайте сложные задачи на более простые подзадачи**

Так же, как в программной инженерии является хорошей практикой декомпозиция сложной системы на набор модульных компонентов, то же самое верно и для задач, представленных языковой модели. Сложные задачи имеют тенденцию к более высокому проценту ошибок, чем более простые задачи. Кроме того, сложные задачи часто могут быть переопределены как рабочий процесс более простых задач, в котором выходные данные предыдущих задач используются для конструирования входных данных последующих задач.

Тактики:

- Используйте классификацию намерений чтобы идентифицировать наиболее релевантные инструкции для пользовательского запроса
- Для приложений диалогового интерфейса, которые требуют очень длинных разговоров, делайте резюме или фильтруйте предыдущий диалог
- Подводите итоги длинных документов по частям и рекурсивно создавайте полное резюме

### **Дайте модели время "подумать"**

Если вас спросят умножить 17 на 28, возможно, вы не знаете ответ сразу, но можете вывести его со временем. Аналогичным образом модели делают больше ошибок в рассуждениях, когда пытаются ответить сразу, вместо того, чтобы взять время для выработки ответа. Запрос на «цепочку мысли» перед ответом может помочь модели надёжнее продвигаться к правильным ответам.

Тактики:

- Инструктируйте модель сначала выработать собственное решение, прежде чем спешить к выводу
- Используйте внутренний монолог или последовательность запросов, чтобы скрыть процесс рассуждения модели
- Спросите модель, не пропустила ли она что-то при предыдущих проходах

### **Используйте внешние инструменты**

Компенсируйте слабости модели, используя выходные данные других инструментов. Например, система извлечения текста (иногда называемая RAG или retrieval augmented generation) может рассказать модели о релевантных документах. Механизм выполнения кода, такой как Интерпретатор Кода OpenAI, может помочь модели решать математические задачи и выполнять код. Если задача может быть выполнена более надежно или эффективно с помощью инструмента, а не языковой моделью, поручите её инструменту, чтобы получить лучшее из обоих.

Тактики:

- Используйте поиск на основе embeddings для эффективного извлечения знаний
- Используйте выполнение кода для более точных вычислений или вызова внешних API
- Дайте модели доступ к определенным функциям

### **Систематически тестируйте изменения**

Улучшение производительности становится проще, если вы можете измерить ее. В некоторых случаях изменение в prompt может привести к лучшей производительности на нескольких изолированных примерах, но привести к худшей общей производительности на более репрезентативном наборе примеров. Поэтому, чтобы быть уверенным в том, что изменение положительно сказывается на производительности, может потребоваться определить комплексный набор тестов (также известный как "eval").

Тактика:

- Оценивайте выходные данные модели с учетом эталонных ответов

### **Тактики**

Каждую из перечисленных стратегий можно реализовать с помощью конкретных тактик. Эти тактики предлагаются в качестве идей для попыток. Они ни в коем случае не являются полностью исчерпывающими, и вы должны чувствовать себя свободно, пробуя творческие идеи, не представленные здесь.

**Стратегия: Пишите четкие инструкции**

**Тактика: Включите детали в ваш запрос, чтобы получить более релевантные ответы**

Чтобы получить максимально релевантный ответ, убедитесь, что запросы содержат все важные детали или контекст. В противном случае вы оставляете модели угадывать, что вы имеете в виду.

<b>Хуже</b>	<b>Лучше</b>
Как мне сложить числа в Excel?	Как мне сложить ряд сумм в долларах в Excel? Я хочу сделать это автоматически для всего листа строк, и чтобы общие итоги оказывались справа в колонке, названной "Итого".
Кто президент?	Кто был президентом Мексики в 2021 году, и как часто проводятся выборы?
Напишите код для расчета последовательности Фибоначчи.	Напишите функцию на TypeScript для эффективного вычисления последовательности Фибоначчи. Обильно комментируйте код, чтобы объяснить, что делает каждая часть и почему она написана таким образом.
Сделайте резюме заметок совещания.	Составьте резюме заметок совещания в одном абзаце. Затем напишите в markdown список докладчиков и каждого из их ключевых пунктов. Наконец, перечислите следующие шаги или предложенные действия докладчиками, если таковые имеются.

### **Тактика: Попросите модель принять персонажа**

Системное сообщение может использоваться, чтобы указать персонажа, используемого моделью в своих ответах.

**SYSTEM**

Когда я прошу помощь в написании чего-либо, вы будете отвечать документом, который содержит хотя бы одну шутку или игривый комментарий в каждом абзаце.

**USER**

Напишите благодарственное письмо моему поставщику стальных болтов за то, что доставка была выполнена вовремя и в краткий срок. Это позволило нам выполнить важный заказ.

### **Тактика: Используйте разделители для четкого обозначения различных частей ввода**

Разделители, такие как тройные кавычки, XML-теги, заголовки разделов и т.д., могут помочь разграничивать текст на секции, которые будут обрабатываться по-разному.

**USER**

Сделайте резюме текста, ограниченного тройными кавычками, в форме хайку.

""""вставьте текст здесь""""

**SYSTEM**

Вам будут предоставлены две статьи (ограниченные XML-тегами) по одной теме. Сначала сделайте резюме аргументов каждой из статей. Затем укажите, какая из них представляет более убедительный аргумент и объясните почему.

**USER**

<article> вставьте первую статью здесь </article>

<article> вставьте вторую статью здесь </article>

**SYSTEM**

Вам будет предоставлено резюме диссертации и предложенное название для неё. Название диссертации должно давать читателю хорошее представление о теме диссертации, но также должно быть привлекательным. Если название не соответствует этим критериям, предложите 5 альтернатив.

**USER**

Резюме: вставьте резюме здесь

Название: вставьте название здесь

Для простых задач, таких как эти, использование разделителей может не сказаться на качестве результата. Однако, чем сложнее задача, тем важнее четко разграничивать детали задачи. Не заставляйте модель работать, чтобы понять, что именно вы от нее требуете.

**Тактика: Укажите шаги, необходимые для выполнения задачи**

Некоторые задачи лучше всего описывать последовательностью шагов. Явное описание шагов может облегчить модели их выполнение.

**SYSTEM**

Используйте следующие пошаговые инструкции для ответа на запросы пользователя.

Шаг 1 - Пользователь предоставит вам текст в тройных кавычках. Сведите этот текст к одному предложению с префиксом "Сводка: ".

Шаг 2 - Переведите сводку из Шага 1 на испанский язык, с префиксом "Перевод: ".

USER

""вставьте текст здесь""

### **Тактика: Предоставление примеров**

Предоставление общих инструкций, применимых ко всем примерам, как правило, более эффективно, чем демонстрация всех возможных вариаций задачи на примере, но в некоторых случаях предоставление примеров может быть проще. Например, если вы хотите, чтобы модель копировала определенный стиль ответов на запросы пользователя, который сложно описать явно. Это известно как подход с использованием "few-shot" запросов.

SYSTEM

Отвечайте в постоянном стиле.

USER

Научи меня терпению.

ASSISTANT

Река, которая вырывает самые глубокие долины, берет начало из скромного источника; величайшая симфония зарождается из одной ноты; самая сложная гобелен начинается с одной нити.

USER

Научи меня океану.

### **Тактика: Укажите желаемую длину вывода**

Вы можете попросить модель создать выводы заданной целевой длины. Целевая длина вывода может быть указана в терминах количества слов, предложений, абзацев, пунктов списка и т.д. Однако следует отметить, что предписание модели генерировать конкретное количество слов работает не с высокой точностью. Модель может более надежно генерировать выводы с определенным количеством абзацев или пунктов списка.

USER

Сделайте краткое изложение текста в рамках тройных кавычек примерно в 50 словах.

""вставьте текст здесь""

USER

Сделайте краткое изложение текста в рамках тройных кавычек в 2 абзаца.

""вставьте текст здесь""

USER

Сделайте краткое изложение текста в рамках тройных кавычек в 3 пункта списка.

""""вставьте текст здесь""""

### **Стратегия: Предоставьте справочный текст**

#### **Тактика: Инструктируйте модель ответить, используя справочный текст**

Если мы можем предоставить модели доверенную информацию, которая актуальна для текущего запроса, то мы можем инструктировать модель использовать предоставленную информацию для составления своего ответа.

SYSTEM

Используйте предоставленные статьи в рамках тройных кавычек для ответа на вопросы. Если ответ невозможно найти в статьях, напишите "Я не смог найти ответ."

USER

<вставьте статьи, каждую в рамках тройных кавычек>

Вопрос: <вставьте вопрос здесь>

Учитывая, что все модели имеют ограниченные окна контекста, нам нужен способ динамически искать информацию, которая актуальна для задаваемого вопроса. Embeddings могут использоваться для реализации эффективного поиска знаний. Смотрите тактику "Используйте поиск на основе embeddings для реализации эффективного поиска знаний" для получения более подробной информации о том, как это реализовать.

#### **Тактика: Инструктировать модель отвечать с цитированием из источника**

Тактика: Инструктировать модель отвечать с цитированием из источника

SYSTEM

Вам будет предоставлен документ, ограниченный тройными кавычками, и вопрос. Ваша задача - ответить на вопрос, используя только предоставленный документ, и цитировать проход(ы) документа, использованные для ответа на вопрос. Если документ не содержит информации, необходимой для ответа на этот вопрос, то просто напишите: "Недостаточно информации". Если на вопрос предоставлен ответ, он должен быть аннотирован с цитированием. Используйте следующий формат для цитирования соответствующих проходов ({"citation": ...}).

USER

""<вставьте документ здесь>""

Вопрос: <вставьте вопрос здесь>

## **Стратегия: Разбивайте сложные задачи на более простые подзадачи**

### **Тактика: Использовать классификацию намерений для определения наиболее релевантных инструкций для запроса пользователя**

Для задач, в которых требуется множество независимых наборов инструкций для обработки различных случаев, может быть полезно сначала классифицировать тип запроса и использовать эту классификацию для определения того, какие инструкции необходимы. Это может быть достигнуто путем определения фиксированных категорий и жесткого кодирования инструкций, которые актуальны для обработки задач в данной категории. Этот процесс также может быть применен рекурсивно для разбиения задачи на последовательность этапов. Преимущество этого подхода заключается в том, что каждый запрос будет содержать только те инструкции, которые необходимы для выполнения следующего этапа задачи, что может привести к более низким показателям ошибок по сравнению с использованием одного запроса для выполнения всей задачи. Это также может привести к снижению затрат, поскольку большие запросы стоят дороже (смотрите информацию о ценах).

Предположим, например, что для приложения службы поддержки клиентов запросы могут быть полезно классифицированы следующим образом:

#### **SYSTEM**

Вам будут предоставлены запросы службы поддержки клиентов. Классифицируйте каждый запрос на первичную и вторичную категории. Предоставьте ваш выход в формате json с ключами: primary и secondary.

Основные категории: Биллинг, Техническая поддержка, Управление аккаунтом или Общий запрос.

Вторичные категории биллинга:

- Отписаться или повысить
- Добавить способ оплаты
- Объяснение по начислениям
- Оспорить начисление

Вторичные категории технической поддержки:

- Устранение неполадок
- Совместимость устройства
- Обновления программного обеспечения

Вторичные категории управления аккаунтом:

- Сброс пароля
- Обновить персональную информацию
- Закрывать аккаунт
- Безопасность аккаунта

Вторичные категории общих запросов:

- Информация о продукте
- Ценообразование
- Обратная связь
- Разговор с оператором

USER

Мне нужно восстановить работу интернета.

На основании классификации запроса клиента, модели можно предоставить более конкретные инструкции для дальнейших действий. Например, предположим, что клиенту требуется помощь в "устранении неполадок".

SYSTEM

Вам будут предоставлены запросы службы поддержки клиентов, требующие устранения неполадок в контексте технической поддержки. Помогите пользователю, выполните следующее:

- Попросите проверить, все ли кабели от/к маршрутизатору подключены. Заметьте, что со временем кабели могут ослабнуть.
- Если все кабели подключены и проблема не устранена, спросите, какую модель маршрутизатора они используют
- Теперь вы посоветуете им, как перезапустить устройство:
  - Если номер модели MTD-327J, посоветуйте нажать красную кнопку и удерживать ее в течение 5 секунд, затем подождите 5 минут перед тестированием соединения.
  - Если номер модели MTD-327S, посоветуйте вынуть и снова включить его в розетку, затем подождите 5 минут перед тестированием соединения.
- Если проблема клиента не устранится после перезапуска устройства и ожидания 5 минут, соедините их с IT поддержкой, выведя IT support requested.
- Если пользователь начинает задавать вопросы, не связанные с этой темой, то подтвердите, хотят ли они завершить текущий чат об устранении неполадок и классифицируйте их запрос в соответствии со следующей схемой:

<insert primary/secondary classification scheme from above here>

USER

Мне нужно восстановить работу интернета.

Обратите внимание, что модель была настроена на вывод специальных строк для указания на изменение состояния диалога. Это позволяет нам превратить нашу систему в конечный автомат, где состояние определяет, какие инструкции будут введены. Отслеживая состояние, какие инструкции актуальны в данном состоянии, и также опционально, какие переходы состояний разрешены из этого состояния, мы можем создать ограничения для пользовательского опыта, которые трудно достичь с менее структурированным подходом.

### **Тактика: Для приложений диалога, требующих очень длинных разговоров, суммируйте или фильтруйте предыдущий диалог**

Поскольку модели имеют фиксированную длину контекста, диалог между пользователем и помощником, в который включен весь разговор в окно контекста, не может продолжаться бесконечно.

Существуют различные решения этой проблемы, одним из которых является суммирование предыдущих оборотов диалога. Как только размер ввода достигает predetermined пороговой длины, это может запустить запрос, который суммирует часть диалога, и суммарная информация предыдущего диалога может быть включена в часть системного сообщения. В качестве альтернативы, предыдущий диалог может быть суммирован асинхронно в фоновом режиме на протяжении всего разговора.

Альтернативное решение - динамически выбирать предыдущие части диалога, которые наиболее актуальны для текущего запроса. См. тактику "Использовать поиск на основе embeddings для эффективного извлечения знаний".

### **Тактика: Суммировать длинные документы по частям и рекурсивно создавать полное резюме**

Поскольку модели имеют фиксированную длину контекста, они не могут быть использованы для суммирования текста, длиннее чем длина контекста минус длина созданного резюме, за один запрос.

Чтобы суммировать очень длинный документ, такой как книга, мы можем использовать последовательность запросов для суммирования каждого раздела документа. Резюме разделов могут быть соединены и суммированы, создавая резюме резюме. Этот процесс может продолжаться рекурсивно до тех пор, пока не будет суммирован весь документ. Если необходимо использовать информацию о предыдущих

разделах, чтобы понять последующие разделы, то дополнительный прием, который может быть полезен, - это включение непрерывного резюме текста, предшествующего любой данной точке книги при суммировании содержимого на этой точке. Эффективность этой процедуры для суммирования книг была исследована в предыдущих исследованиях OpenAI с использованием вариантов GPT-3.

## **Стратегия: Дать моделям время на "размышления"**

### **Тактика: Инструктируйте модель разработать собственное решение, прежде чем спешить к выводу**

Иногда мы получаем лучшие результаты, когда мы явно инструктируем модель рассуждать с первопринципов, перед тем как прийти к выводу. Предположим, например, мы хотим, чтобы модель оценила решение ученика по математической задаче. Самый очевидный способ подхода - просто спросить модель, правильно ли решение ученика или нет.

SYSTEM

Определите, правильно ли решение ученика или нет.

USER

Условие задачи: Я строю установку солнечной энергии и мне нужна помощь в расчёте финансов.

- Стоимость земли \$100 / за квадратный фут
  - Я могу купить солнечные панели по \$250 / за квадратный фут
  - Я заключил контракт на обслуживание, который будет стоить мне фиксированные \$100k в год, и дополнительно \$10 / за квадратный фут
- Какова общая стоимость за первый год эксплуатации в зависимости от количества квадратных футов.

Решение ученика: Пусть  $x$  будет размером установки в квадратных футах.

1. Стоимость земли:  $100x$

2. Стоимость солнечных панелей:  $250x$

3. Стоимость обслуживания:  $100,000 + 100x$

Общая стоимость:  $100x + 250x + 100,000 + 100x = 450x + 100,000$

ASSISTANT

Решение ученика правильное.

Но решение ученика на самом деле неверное! Мы можем добиться чтобы модель успешно заметила это, попросив модель сгенерировать собственное решение в первую очередь.

SYSTEM

Сначала разработайте свое собственное решение задачи. Затем сравните ваше решение с решением ученика и оцените, правильно ли решение ученика или нет. Не решайте, правильно ли решение ученика, пока не решите задачу сами.

USER

Условие задачи: Я строю установку солнечной энергии и мне нужна помощь в расчёте финансов.

- Стоимость земли \$100 / за квадратный фут
  - Я могу купить солнечные панели по \$250 / за квадратный фут
  - Я заключил контракт на обслуживание, который будет стоить мне фиксированные \$100к в год, и дополнительно \$10 / за квадратный фут
- Какова общая стоимость за первый год эксплуатации в зависимости от количества квадратных футов.

Решение ученика: Пусть  $x$  будет размером установки в квадратных футах.

1. Стоимость земли:  $100x$
  2. Стоимость солнечных панелей:  $250x$
  3. Стоимость обслуживания:  $100,000 + 100x$
- Общая стоимость:  $100x + 250x + 100,000 + 100x = 450x + 100,000$

ASSISTANT

Пусть  $x$  будет размером установки в квадратных футах.

1. Стоимость земли:  $100x$
  2. Стоимость солнечных панелей:  $250x$
  3. Стоимость обслуживания:  $100,000 + 10x$
- Общая стоимость:  $100x + 250x + 100,000 + 10x = 360x + 100,000$

Решение ученика неверное. Они допустили ошибку в расчете стоимости обслуживания, используя  $100x$  вместо  $10x$ . Правильная общая стоимость за первый год эксплуатации составляет  $360x + 100,000$ .

### **Тактика: Использование внутреннего монолога или последовательности запросов для скрытия процесса рассуждения модели**

Преыдушая тактика показывает, что иногда важно, чтобы модель подробно рассуждала о проблеме перед тем, как ответить на конкретный вопрос. Для некоторых приложений процесс рассуждения, который модель использует для выхода на окончательный ответ, был бы неуместно делиться с пользователем. Например, в приложениях для обучения мы можем хотеть побудить студентов самостоятельно решить задачу, но процесс рассуждения модели о решении студента может раскрыть ответ студенту.

Внутренний монолог - это тактика, которую можно использовать для смягчения этого. Идея внутреннего монолога заключается в том, чтобы указать модели поместить части вывода, которые должны быть скрыты от пользователя, в структурированный формат, который облегчает их анализ. Затем перед тем, как представить вывод пользователю, вывод анализируется, и видимой делается только часть вывода.

SYSTEM

Следуйте этим шагам, чтобы ответить на запросы пользователя.

Шаг 1 - Сначала решите задачу самостоятельно. Не полагайтесь на решение студента, так как оно может быть неверным. Заключите всю вашу работу на этом шаге в тройные кавычки (""").

Шаг 2 - Сравните ваше решение с решением студента и оцените, правильно ли решение студента. Заключите всю вашу работу на этом шаге в тройные кавычки (""").

Шаг 3 - Если студент допустил ошибку, определите, какую подсказку вы могли бы дать студенту, не раскрывая ответ. Заключите всю вашу работу на этом шаге в тройные кавычки (""").

Шаг 4 - Если студент допустил ошибку, предоставьте студенту подсказку из предыдущего шага (вне тройных кавычек). Вместо того, чтобы писать "Шаг 4 - ..." пишите "Подсказка:".

USER

Формулировка задачи: <insert problem statement>

Решение студента: <insert student solution>

Также это можно достичь с помощью последовательности запросов, в которых все, кроме последнего, имеют скрытый вывод для конечного пользователя.

Сначала мы можем попросить модель решить задачу самостоятельно. Поскольку этот начальный запрос не требует решения ученика, его можно опустить. Это дает дополнительное преимущество, заключающееся в том, что нет шанса, что решение модели будет искажено попыткой решения ученика.

USER

<вставьте условие задачи>

Далее, мы можем позволить модели использовать всю доступную информацию для оценки правильности решения ученика.

SYSTEM

Сравните ваше решение с решением ученика и оцените, правильно ли решение ученика.

USER

Условие задачи: """"<вставьте условие задачи>""""

Ваше решение: """"<вставьте решение, сгенерированное моделью>""""

Решение ученика: """"<вставьте решение ученика>""""

Наконец, мы можем позволить модели использовать свой собственный анализ для создания ответа в роли полезного репетитора.

SYSTEM

Вы — репетитор по математике. Если ученик допустил ошибку, предложите ему подсказку таким образом, чтобы не раскрыть ответ. Если ученик не допустил ошибки, просто предложите ему ободряющий комментарий.

USER

Условие задачи: ""<вставьте условие задачи>""

Ваше решение: ""<вставьте решение, сгенерированное моделью>""

Решение ученика: ""<вставьте решение ученика>""

Анализ: ""<вставьте анализ, сгенерированный моделью на предыдущем шаге>""

### **Тактика: Спросите модель, пропустила ли она что-то на предыдущих проходах**

Представим, что мы используем модель для перечисления отрывков из источника, которые релевантны определённому вопросу. После перечисления каждого отрывка модель должна определить, должна ли она начать писать следующий отрывок или должна остановиться. Если исходный документ большой, обычно модель останавливается слишком рано и не перечисляет все релевантные отрывки. В таком случае, лучшую производительность часто можно получить, задавая модели дополнительные вопросы, чтобы найти любые отрывки, которые она пропустила на предыдущих проходах.

SYSTEM

Вам будет предоставлен документ, ограниченный тройными кавычками. Ваша задача - выбрать отрывки, которые относятся к следующему вопросу: "Какие значимые сдвиги парадигм произошли в истории искусственного интеллекта".

Убедитесь, что отрывки содержат весь релевантный контекст, необходимый для их интерпретации - другими словами, не извлекайте маленькие фрагменты, в которых отсутствует важный контекст.

Предоставьте вывод в формате JSON следующим образом:

```
["excerpt": "..."],
```

```
...
```

```
["excerpt": "..."]]
```

USER

```
""<insert document here>""
```

ASSISTANT

```
[{"excerpt": "здесь модель пишет отрывок"},
```

```
...
```

```
{"excerpt": "здесь модель пишет еще один отрывок"}]
```

USER

Есть ли ещё релевантные отрывки? Старайтесь не повторять отрывки. Также убедитесь, что отрывки содержат весь релевантный контекст, необходимый для их интерпретации - другими словами, не извлекайте маленькие фрагменты, в которых отсутствует важный контекст.

## **Стратегия: Использование внешних инструментов**

### **Тактика: Использование поиска на базе embeddings для эффективного извлечения знаний**

Модель может использовать внешние источники информации, если они предоставлены в качестве части её входных данных. Это может помочь модели генерировать более информированные и актуальные ответы. Например, если пользователь задает вопрос о конкретном фильме, может быть полезно добавить высококачественную информацию о фильме (например, актеры, режиссер и т.д.) во входные данные модели. Embeddings могут использоваться для реализации эффективного извлечения знаний, чтобы соответствующая информация могла быть добавлена к входным данным модели динамично во время выполнения.

Текстовый embedding - это вектор, который может измерять степень схожести между текстовыми строками. Похожие или релевантные строки будут ближе друг к другу, чем несвязанные строки. Этот факт, наряду с существованием быстрых алгоритмов поиска векторов, означает, что embeddings могут использоваться для реализации эффективного извлечения знаний. В частности, текстовый корпус может быть разделен на части, и каждая часть может быть представлена в виде embedding и храниться. Затем может быть представлен запрос в виде embedding, и может быть выполнен поиск вектора для нахождения вложенных частей текста из корпуса, которые наиболее связаны с запросом (то есть наиболее близки в пространстве embeddings).

Смотрите тактику "Инструктировать модель использовать извлеченные знания для ответа на запросы" для примера того, как использовать извлечение знаний для минимизации вероятности того, что модель представит неверные факты.

### **Тактика: Использование выполнения кода для выполнения более точных расчетов или вызова внешних API**

На языковые модели нельзя полагаться при выполнении арифметических или длинных расчетов точно самостоятельно. В

случаях, когда это необходимо, модель может быть настроена на написание и выполнение кода вместо собственных расчетов. В частности, модель может быть настроена на размещение кода, который должен быть выполнен, в определенном формате, таком как тройные обратные кавычки. После получения вывода код может быть извлечен и выполнен. Наконец, если это необходимо, вывод из среды выполнения кода (например, интерпретатора Python) может быть представлен в качестве входных данных для модели для следующего запроса.

SYSTEM

Вы можете писать и выполнять код Python, заключив его в тройные обратные кавычки, например, ```здесь код```. Используйте это для выполнения расчетов.

USER

Найти все действительные корни следующего полинома:  $3x^5 - 5x^4 - 3x^3 - 7x - 10$ .

Еще один хороший пример использования выполнения кода - вызов внешних API. Если модель инструктирована о правильном использовании API, она может написать код, который использует его. Модель может быть инструктирована о том, как использовать API, предоставив ей документацию и/или примеры кода, показывающие, как использовать API.

SYSTEM

Вы можете писать и выполнять код Python, заключив его в тройные обратные кавычки. Также обратите внимание, что у вас есть доступ к следующему модулю, чтобы помочь пользователям отправлять сообщения своим друзьям:

```
```python
import message
message.write(to="John", message="Привет, встретимся после
работы?")```
```

**ВНИМАНИЕ:** Выполнение кода, созданного моделью, не является по своей сути безопасным, и при любом использовании, которое стремится сделать это, следует предпринять меры предосторожности. В частности, требуется среда выполнения кода с ограничением, чтобы ограничить вред, который могут нанести ненадежные коды.

**Тактика:** Предоставьте модели доступ к определенным функциям

API завершения чата позволяет передавать список описаний функций в запросах. Это позволяет моделям генерировать аргументы функций в соответствии с предоставленными схемами. Сгенерированные аргументы функций возвращаются API в формате JSON и могут быть

использованы для выполнения вызовов функций. Вывод, предоставленный вызовами функций, затем может быть возвращен в модель в следующем запросе для замыкания цикла. Это рекомендуемый способ использования моделей OpenAI для вызова внешних функций.

## Стратегия: Систематически тестируйте изменения

Иногда может быть сложно определить, делает ли изменение — например, новая инструкция или новый дизайн — вашу систему лучше или хуже. Просмотр нескольких примеров может намекнуть на то, что лучше, но при малых размерах выборки может быть сложно отличить настоящее улучшение от случайной удачи. Возможно, изменение улучшает производительность на некоторых входах, но ухудшает на других.

Процедуры оценки (или "evals") полезны для оптимизации дизайна систем. Хорошие evals:

- Репрезентативны для использования в реальном мире (или по крайней мере разнообразны)
- Содержат множество тестовых случаев для большей статистической мощности (см. таблицу ниже для рекомендаций)
- Легко автоматизировать или повторять

РАЗЛИЧИЕ ДЛЯ ОБНАРУЖЕНИЯ	НЕОБХОДИМЫЙ РАЗМЕР ВЫБОРКИ ДЛЯ 95% УВЕРЕННОСТИ
--------------------------	------------------------------------------------

30%	~10
10%	~100
3%	~1,000
1%	~10,000

Оценка выходных данных может выполняться компьютерами, людьми или их сочетанием. Компьютеры могут автоматизировать evals с объективными критериями (например, вопросы с одним правильным ответом), а также с некоторыми субъективными или нечеткими критериями, при которых выходные данные модели оцениваются другими запросами модели. OpenAI Evals — это программа с открытым исходным кодом, предоставляющая инструменты для создания автоматизированных evals.

Модельные evals могут быть полезны, когда существует ряд возможных выходных данных, которые будут считаться равноценными по качеству (например, для вопросов с длинными ответами). Граница между тем, что можно реалистично оценить с помощью модельного

eval и что требует оценки человеком, нечетка и постоянно сдвигается по мере того, как модели становятся более способными. Мы рекомендуем экспериментировать, чтобы понять, насколько хорошо модельные evals могут работать для вашего случая использования.

### **Тактика: Оценка выходных данных модели с учетом эталонных ответов**

Предположим, что известно, что правильный ответ на вопрос должен ссылаться на конкретный набор известных фактов. Тогда мы можем использовать запрос модели, чтобы подсчитать, сколько требуемых фактов содержится в ответе.

Например, используя следующее системное сообщение:

**SYSTEM**

Вам будет предоставлен текст, ограниченный тройными кавычками, который должен быть ответом на вопрос. Проверьте, содержатся ли в ответе следующие сведения:

- Нил Армстронг был первым человеком, ступившим на Луну.
- Дата первой прогулки Нила Армстронга по Луне была 21 июля 1969 года.

Для каждого из этих пунктов выполните следующие шаги:

- 1 - Переформулируйте пункт.
- 2 - Приведите цитату из ответа, которая наиболее близка к этому пункту.
- 3 - Рассмотрите, мог ли человек, читающий цитату и не знакомый с темой, сразу сделать вывод о пункте. Объясните, почему да или почему нет, прежде чем принимать решение.
- 4 - Напишите "да", если ответ на пункт 3 был утвердительный, в противном случае напишите "нет".

В заключение подсчитайте, сколько ответов "да" получено. Укажите это количество как {"count": <вставьте сюда количество>}.

Вот пример ввода, где оба пункта удовлетворены:

**SYSTEM**

<вставьте сообщение системы выше>

**USER**

""Нил Армстронг знаменит тем, что стал первым человеком, ступившим на Луну. Это историческое событие произошло 21 июля 1969 года в ходе миссии "Аполлон-11".""

Вот пример ввода, где удовлетворен только один пункт:

SYSTEM

<вставьте сообщение системы выше>

USER

""Нил Армстронг вошел в историю, когда вышел из лунного модуля, став первым человеком, ступившим на Луну.""

SYSTEM

<вставьте сообщение системы выше>

USER

""Летом 69-го, великое путешествие, Аполлон 11, смело, как легенда. Армстронг сделал шаг, история раскрылась, "Один маленький шаг," сказал он, для нового мира.""

Есть много возможных вариантов на эту тему модельной оценки. Рассмотрим следующую разновидность, которая отслеживает тип перекрытия между кандидатом на ответ и эталонным ответом, а также отслеживает, противоречит ли кандидат на ответ какой-либо части эталонного ответа.

SYSTEM

Используйте следующие шаги для ответа на входные данные пользователя. Полностью повторяйте каждый шаг перед продолжением. т.е. "Шаг 1: Рассуждение...".

Шаг 1: Подумайте шаг за шагом, является ли информация в представленном ответе по сравнению с ответом эксперта: несвязанными, равными, подмножеством, надмножеством или перекрывающимися (т.е. некоторое пересечение, но не подмножество/надмножество).

Шаг 2: Подумайте шаг за шагом, противоречит ли представленный ответ какому-либо аспекту ответа эксперта.

Шаг 3: Выведите объект JSON, структурированный так: {"type\_of\_overlap": "несвязанные" или "равные" или "подмножество" или "надмножество" или "перекрывающиеся", "contradiction": true или false}

Вот пример входных данных с неудовлетворительным ответом, который тем не менее не противоречит ответу эксперта:

SYSTEM

<вставьте сообщение системы выше>

USER

Вопрос: ""За какое событие Нил Армстронг наиболее известен и когда оно произошло? Предполагается время UTC.""

Ответ пользователя: ""Разве он не гулял по Луне или что-то в этом

духе?""

Ответ эксперта: ""Нил Армстронг наиболее известен тем, что стал первым человеком, ступившим на Луну. Это историческое событие произошло 21 июля 1969 года.""

Вот пример ввода с ответом, который прямо противоречит ответу эксперта:

SYSTEM

<insert system message above>

USER

Вопрос: ""За какое событие Нил Армстронг наиболее известен и когда оно произошло? Предполагаем UTC время.""

Предложенный Ответ: ""21 июля 1969 года Нил Армстронг стал вторым человеком, ступившим на поверхность Луны, следуя за Баззом Олдрином.""

Ответ Эксперта: ""Нил Армстронг наиболее известен тем, что стал первым человеком, ступившим на Луну. Это историческое событие произошло 21 июля 1969 года.""

Вот пример ввода с правильным ответом, который также содержит немного более подробную информацию, чем необходимо:

SYSTEM

<insert system message above>

USER

Вопрос: ""За какое событие Нил Армстронг наиболее известен и когда оно произошло? Предполагаем UTC время.""

Предложенный Ответ: ""Примерно в 02:56 UTC 21 июля 1969 года Нил Армстронг стал первым человеком, ступившим на поверхность Луны, что стало монументальным достижением в истории человечества.""

Ответ Эксперта: ""Нил Армстронг наиболее известен тем, что стал первым человеком, ступившим на Луну. Это историческое событие произошло 21 июля 1969 года.""