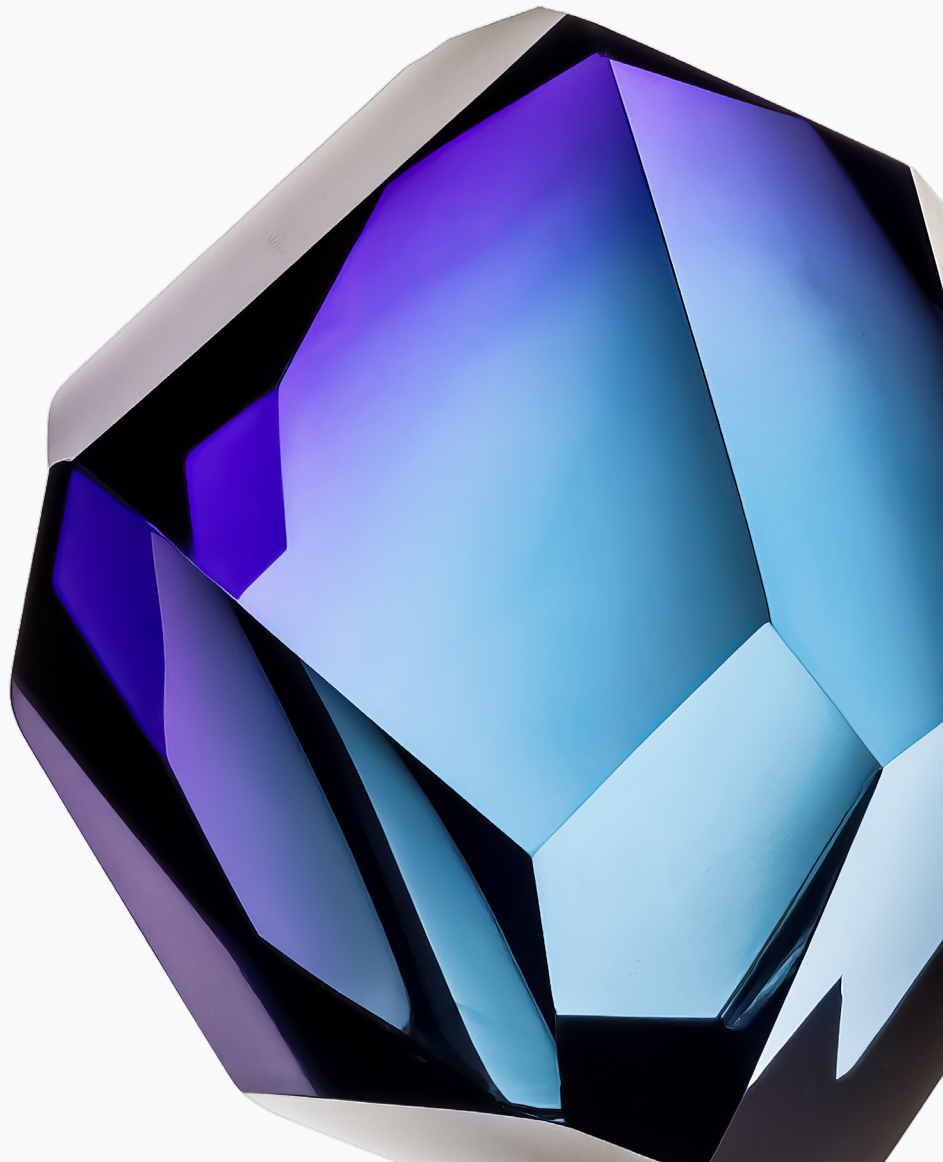


# Prompt Engineering

Author: Lee Boonstra



Prompt Engineering

## Благодарности

### Составители контента

Майкл Шерман

Юань Цао

Эрик Армбруст

Анант Навалгария

Антонио Гулли

Симон Каммель

### Кураторы и редакторы

Антонио Гулли

Анант Навалгария

Грейс Моллисон

### Технический писатель

Джоуи Хеймейкер

### Дизайнер

Майкл Лэннинг


Февраль 2025 г.

главле

Введение	6
Оперативное проектирование	7
Конфигурация выхода LLM	8
Длина выходного отверстия	8
Элементы управления отбором проб	9
Температура	9
Топ-К и топ-П	10
Собираем все вместе	11
Приемы подталкивания	13
Общие вопросы / нулевой выстрел	13
Один выстрел и несколько выстрелов	15
Системные, контекстные и ролевые подсказки	18
Системные подсказки	19
Ролевые подсказки	21
Контекстная подсказка	23

Пошаговая подсказка .....	25
Цепь мыслей (CoT) .....	29
Самосогласованность .....	32
Древо мыслей (ToT) .....	36
ReAct (reason & act) .....	37
Автоматическое проектирование .....	40
Код подсказки .....	42
Подсказки для написания кода .....	42
Подсказки для объяснения кода .....	44
Подсказки для перевода кода .....	46
Подсказки для отладки и проверки кода .....	48
Как насчет мультимодальных подсказок? .....	54
Лучшие практики .....	54
Приведите примеры .....	54
Дизайн с простотой .....	55
Укажите конкретные результаты .....	56
Используйте инструкции вместо ограничений .....	56
Контроль максимальной длины токена .....	58
Использование переменных в подсказках .....	58
Экспериментируйте с форматами ввода и стилями письма .....	59
Для получения нескольких подсказок с заданиями на классификацию смешивайте классы .....	59
Адаптация к обновлениям модели .....	60
Экспериментируйте с форматами вывода .....	60

Восстановление JSON	61
Работа со схемами	62
Экспериментируйте вместе с другими оперативными инженерами	63
Передовой опыт CoT	64
Документируйте различные попытки подсказки	64
Резюме	66
Примечания	68



Вам не нужно быть специалистом по изучению данных или инженером машинного обучения - написать подсказку может каждый.

## Введение

Если говорить о входе и выходе большой языковой модели, то текстовая подсказка (иногда сопровождаемая другими модальностями, например, подсказками в виде изображений) - это вход, который использует модель чтобы предсказать конкретный результат. Вам не нужно быть специалистом по изучению данных или инженером машинного обучения - написать подсказку может каждый. Однако создать наиболее эффективную подсказку может быть непросто. На эффективность подсказки влияют многие аспекты: используемая модель, обучающие данные модели, конфигурации модели, выбор слов, стиль и тон, структура и контекст. Поэтому разработка подсказок - это итеративный процесс. Неадекватные подсказки могут привести к двусмысленным и неточным ответам и помешать модели выдать значимый результат.

Когда вы общаетесь с чатботом Gemini,<sup>1</sup> вы, по сути, пишете подсказки, однако в этом обзоре речь идет о написании подсказок для модели Gemini внутри Vefiex AI или с помощью API, поскольку, обращаясь к модели напрямую, вы получаете доступ к ее конфигурации, такой как температура и т. д.

В этом документе подробно рассматривается техника подсказок. Мы рассмотрим различные техники создания подсказок, которые помогут вам стать стаффом, а также поделимся советами и лучшими практиками, которые помогут вам стать экспертом в области создания подсказок. Мы также обсудим некоторые проблемы, с которыми вы можете столкнуться при создании подсказок.

## Оперативное проектирование

Вспомните, как работает LLM: это механизм предсказания. Модель получает на вход последовательный текст, а затем предсказывает, какой должна быть следующая лексема, основываясь на данных, на которых она обучалась. LLM работает таким образом, чтобы делать это снова и снова, добавляя ранее предсказанную лексему в конец последовательного текста для предсказания следующей лексемы. Следующий

Предсказание лексем основано на связи между тем, что содержится в предыдущих лексемах, и тем, что LLM видел во время обучения.

Когда вы пишете подсказку, вы пытаетесь настроить LLM на предсказание правильной последовательности лексем. Проектирование подсказок - это процесс разработки высококачественных подсказок, которые направляют LLM на получение точных результатов. Этот процесс включает в себя поиск лучшей подсказки, оптимизацию длины подсказки и оценку стиля и структуры подсказки в зависимости от того, как она написана.

для решения задачи. В контексте обработки естественного языка и LLM подсказка - это входные данные, предоставляемые модели для генерации ответа или предсказания.

Эти подсказки могут использоваться для решения различных задач понимания и генерации, таких как обобщение текста, извлечение информации, вопросы и ответы, классификация текста, перевод языка или кода, генерация кода, документирование кода или аргументация.

Пожалуйста, не стесняйтесь обращаться к руководствам по подсказкам Google<sup>2,3</sup> с простыми и эффективными примерами подсказок.

При разработке подсказок вы начнете с выбора модели. Подсказки могут потребовать оптимизации под конкретную модель, независимо от того, используете ли вы языковые модели Gemini в Vertex AI, GPT, Claude или модели с открытым исходным кодом, такие как Gemma или LLaMA.

Помимо подсказок, вам также придется иметь дело с различными конфигурациями LLM.

## Конфигурация выхода LLM

После выбора модели вам нужно будет определиться с ее конфигурацией. Большинство LLM поставляются с различными опциями конфигурации, которые управляют выходными данными LLM. Эффективная разработка оперативной информации требует установки этих конфигураций оптимальным образом для решения вашей задачи.

### Длина выходного отверстия

Важным параметром конфигурации является количество маркеров, генерируемых в ответе. Генерация большого количества маркеров требует от LLM большого количества вычислений, что приводит к повышенному потреблению энергии, потенциально более медленному времени отклика и более высокой стоимости.



Уменьшение длины вывода LLM не приводит к тому, что LLM становится более стилистически или текстуально лаконичным в создаваемом им выводе, оно просто заставляет LLM перестать предсказывать больше лексем по достижении предела. Если для ваших нужд требуется длина вывода shof1, вам также, возможно, потребуется разработать подсказку с учетом этого.

Ограничение длины вывода особенно важно для некоторых техник подсказки LLM, таких как ReAct, когда LLM продолжает выдавать бесполезные лексемы после нужного вам ответа.

Имейте в виду, что генерация большего количества токенов требует от LLM большего количества вычислений, что приводит к повышенному потреблению энергии и потенциально более медленному времени отклика, что ведет к увеличению затрат.

## Контроль отбора проб

ЛЛМ формально не предсказывают одну лексему. Скорее, ЛЛМ предсказывают вероятности того, какой может быть следующая лексема, причем каждая лексема в словаре ЛЛМ получает свою вероятность. Вероятности этих лексем затем сэмплируются, чтобы определить, какой будет следующая произведенная лексема.

Temperature, top-K и top-P - наиболее распространенные параметры конфигурации, которые определяют, как обрабатываются предсказанные вероятности токенов для выбора одного выходного токена.

## Температура

Температура регулирует степень случайности при выборе жетонов. Более низкие температуры подходят для подсказок, которые ожидают более детерминированного ответа, в то время как более высокие температуры могут привести к более разнообразным или неожиданным результатам. Температура 0 (жадное декодирование) - это

Детерминированный: всегда выбирается маркер с наибольшей вероятностью (хотя следует учитывать, что если два маркера имеют одинаковую наибольшую предсказанную вероятность, то в зависимости от того, как реализован тайбрейк, вы не всегда можете получить одинаковый результат с температурой 0).

Температура, близкая к максимальной, как правило, создает больше случайных результатов. И по мере того как температура становится все выше и выше, все токены становятся одинаково вероятными для того, чтобы стать следующим предсказанным токеном.

Температурный контроль Gemini можно понимать так же, как функцию softmax, используемую в машинном обучении. Низкая температура отражает низкую температуру softmax (T), подчеркивая единственную, предпочтительную температуру с высокой точностью. Более высокая температура Gemini похожа на высокую температуру softmax, что позволяет расширить диапазон температур вокруг выбранные настройки более приемлемы. Эта повышенная нецелостность позволяет использовать сценарии, в которых жесткая и точная температура может быть не нужна, например, при экспериментах с творческими продуктами.

## Топ-K и топ-П

Тор-K и тор-Р (также известные как выборка ядра)<sup>4</sup> - это две настройки выборки, используемые в LLM, чтобы ограничить предсказание следующего лексем лексемами с наибольшей предсказанной вероятностью. Подобно температуре, эти параметры выборки контролируют случайность и разнообразие генерируемого текста.

- Выборка **Тор-K** отбирает K наиболее вероятных лексем из предсказанного моделью распределения. Чем выше тор-K, тем более творческим и разнообразным будет вывод модели; чем ниже тор-K, тем более спокойным и фактическим будет вывод модели. Значение тор-K, равное 1, эквивалентно жадному декодированию.

- Выборка **Top-P** отбирает лучшие лексемы, чья кумулятивная вероятность не превышает целевого значения (P). Значения P варьируются от 0 (жадное декодирование) до 1 (все лексемы в словаре LLM).

Лучший способ выбрать между top-K и top-P - это поэкспериментировать с обоими методами (или с обоими вместе) и посмотреть, какой из них даст вам результаты.

## Все вместе

Выбор между top-K, top-P, температурой и количеством генерируемых токенов зависит от конкретного приложения и желаемого результата, и все эти параметры влияют друг на друга. Также важно убедиться в том, что вы понимаете, как выбранная вами модель сочетает в себе различные параметры выборки.

Если температура, top-K и top-P доступны (как в Vefiex Studio), то токены, удовлетворяющие критериям top-K и top-P, становятся кандидатами на следующий предсказанный токен, а затем температура применяется для выборки из лексем, прошедших критерии top-K и top-P. Если доступны только top-K или top-P, поведение такое же, но используется только один параметр top-K или P.

Если температура недоступна, то все лексемы, удовлетворяющие критериям top-K и/или top-P, выбираются случайным образом, чтобы получить одну следующую предсказанную лексему.

При экстремальных настройках одного значения конфигурации выборки эта настройка либо отменяет другие настройки конфигурации, либо становится неактуальной.

- Если вы установите температуру на 0, top-K и top-P станут неважными - наиболее вероятный токен станет следующим предсказанным токеном. Если вы установите очень высокую температуру (выше 1 - как правило, до 10), температура станет неважной, и все токены, которые делают по критериям top-K и/или top-P, затем случайным образом выбирается следующий предсказанный маркер.
- Если задать top-K равным 1, температура и top-P становятся неважными. Только одна лексема проходит по критериям top-K, и эта лексема является следующей предсказанной лексемой. Если задать top-K очень высоким, например, равным размеру словаря LLM, то любая лексема с ненулевой вероятностью стать следующей лексемой будет соответствовать критериям top-K, и ни одна из них не будет выбрана.
- Если вы установите top-P в 0 (или очень маленькое значение), то большинство реализаций LLM-выборок будут рассматривать только наиболее вероятный токен на предмет соответствия критерию top-P, делая температуру и top-K неважными. Если вы установите top-P в 1, то любой токен с ненулевой вероятностью стать следующим токеном будет соответствовать критерию top-P, и ни один из них не будет отобран.

В качестве общей точки стафинга, температура .2, top-P .95 и top-K 30 дадут вам относительно последовательные результаты, которые могут быть творческими, но не чрезмерно. Если вам нужны особенно творческие результаты, попробуйте стафить при температуре .9, top-P .99 и top-K 40. А если вам нужны менее творческие результаты, попробуйте стафить с температурой .1, top-P .9 и top-K 20.

Наконец, если ваша задача всегда имеет один правильный ответ (например, ответ на математическую задачу), ставьте температуру 0.

**ПРИМЕЧАНИЕ:** При большей свободе (более высокая температура, top-K, top-P и выходные лексемы) LLM может генерировать текст, который менее релевантен.

**ПРЕДУПРЕЖДЕНИЕ:** Вы когда-нибудь видели ответ, заканчивающийся большим количеством слов-заполнителей? Это также известно как "ошибка цикла повторения", которая является распространенной проблемой в больших языковых моделях, когда модель застревает в цикле, неоднократно генерируя одно и то же (заполняющее) слово, фразу или структуру предложения, что часто усугубляется неподходящей температурой и top-k/

верхние настройки. Это может происходить как при низких, так и при высоких температурах, хотя и по разным причинам. При низких температурах модель становится слишком детерминированной, жестко придерживаясь пути с наибольшей вероятностью, что может привести к заикливанию, если этот путь повторяет ранее сгенерированный текст. И наоборот, при высоких температурах вывод модели становится чрезмерно случайным, увеличивая вероятность того, что случайно выбранное слово или фраза приведет к предыдущему состоянию, что приведет к заикливанию из-за огромного количества доступных вариантов. В обоих случаях процесс выборки модели "застревает", что приводит к монотонному и бесполезному выводу до тех пор, пока окно вывода не будет заполнено. Решение этой проблемы часто требует тщательного изменения температуры и значений top-k/top-p, чтобы найти оптимальный баланс между детерминизмом и случайностью.

## Техника подсказки

LLM настроены на выполнение инструкций и обучаются на больших объемах данных, чтобы понимать подсказки и генерировать ответы. Однако LLM не совершенны: чем четче текст подсказки, тем лучше LLM предсказывает следующий вероятный текст. Кроме того, специальные методы, использующие преимущества того, как обучаются LLM и как работают LLM, помогут вам получить от LLM соответствующие результаты

Теперь, когда мы поняли, что такое разработка подсказок и что для этого нужно, давайте рассмотрим несколько примеров самых неэффективных техник подсказок.

### Общая подсказка / нулевой выстрел

Подсказка с нулевым выстрелом<sup>5</sup> - это самый простой тип подсказки. Она содержит только описание задачи и некоторый текст, который LLM должен ввести. Этим текстом может быть что угодно: вопрос, фрагмент рассказа или инструкция. Название zero-shot означает "без примеров".

Воспользуемся Vefiex AI Studio (для языка) в Vefiex AI,<sup>6</sup> которая предоставляет игровую площадку для тестирования подсказок. В таблице 1 приведен пример нулевого запроса на классификацию рецензий на фильмы.

Формат таблицы, используемый ниже, - отличный способ документирования подсказок. Ваши подсказки, скорее всего, пройдут множество итераций, прежде чем окажутся в кодовой базе, поэтому очень важно отслеживать работу по разработке подсказок в дисциплинированном, структурированном виде. Подробнее о формате таблицы, важности отслеживания работы над подсказками и процессе разработки подсказок - в разделе "Лучшие практики" далее в этой главе ("Документирование различных попыток создания подсказок").

Температура модели должна быть установлена на низкое число, так как креатив не нужен, и мы используем значения top-K и top-P по умолчанию в gemini-pro, которые фактически отключают обе настройки (см. выше "Конфигурация вывода LLM"). Обратите внимание на сгенерированный вывод. Слова *disturbing* и *masterpiece* должны немного усложнить предсказание, поскольку оба слова используются в одном предложении.

Название	1_1_фильм_классификация		
Модель	Классифицируйте отзывы о фильмах как положительные,		
цели	нейтральные или отрицательные. gemini-pro		
Температура	0.1	Лимит жетонов	5
Тор-K Prompt	N/A	Тор-P	1
	Классифицируйте отзывы о фильме как ПОЗИТИВНЫЕ, НЕЙТРАЛЬНЫЕ или НЕГАТИВНЫЕ. Рецензия: "Она" – это тревожное исследование, показывающее, в каком направлении движется человечество, если позволить искусственному интеллекту продолжать бесконтрольно развиваться. Хотелось бы, чтобы таких фильмов, как этот шедевр, было больше. Чувства:		
Выход	ПОЗИТИВ		

Таблица 1. Пример подсказки с нулевым выстрелом

Когда нулевой выстрел не работает, вы можете предоставить демонстрацию или примеры в подсказке, что приводит к подсказке "одним выстрелом" и "несколькими выстрелами". Общие подсказки / нулевой выстрел

## Один выстрел и несколько выстрелов

При создании подсказок для моделей ИИ полезно приводить примеры. Эти примеры помогут модели понять, о чем вы просите. Примеры особенно полезны, когда вы хотите направить модель к структуре или шаблону вывода `sefiain`.

**Одноразовая подсказка** дает *один-единственный* пример, отсюда и название "одноразовая". Идея заключается в том, что у модели есть пример, которому она может подражать, чтобы наилучшим образом выполнить задание.

**Подсказка из нескольких кадров** <sup>7</sup> предоставляет модели *несколько* примеров. Этот подход показывает модели образец, которому она должна следовать. Идея похожа на одноразовую подсказку, но несколько примеров желаемой модели увеличивают вероятность того, что модель будет следовать этой модели.

Количество примеров, которые вам понадобятся для подсказок в несколько кадров, зависит от нескольких факторов, включая сложность задачи, качество примеров и возможности используемой вами модели генеративного ИИ (gen AI). В качестве общего правила следует использовать не менее трех-пяти примеров для подсказки несколькими выстрелами. Однако для более сложных задач может потребоваться большее количество примеров или меньшее количество из-за ограничения длины входных данных в вашей модели.

В таблице 2 показан пример запроса на несколько снимков. Используем те же настройки конфигурации модели gemini-pro, что и раньше, за исключением увеличения лимита маркеров, чтобы учесть необходимость более длительного ответа.

Цель	Разбор заказов на пиццу в		
Модель	JSON gemini-pro		
Температура	0.1	Лимит жетонов	250
Топ-К	Н/Д	Топ-Р	1
Проспект	<p>Разберите заказ пиццы клиента на корректный JSON:</p> <p>ПРИМЕР:</p> <p>Я хочу маленькую пиццу с сыром, томатным соусом и пепперони. Ответ JSON:</p> <pre> {   "размер": "маленький",   "тип": "normal",   "ингредиенты": ["сыр", "томатный соус", "пепперони"] }</pre> <p>Продолжение на следующей странице...</p>		



Справка	<p>ПРИМЕР:</p> <p>Могу ли я получить большую пиццу с томатным соусом, базиликом и моцареллой?</p> <pre>{   "размер": "большой",   "тип": "normal",   "ингредиенты": [ "томатный соус", "бацел", "моцарелла" ] ] . }</pre> <p>Теперь я хочу большую пиццу, в первой половине которой сыр и моцарелла. А вторая – с томатным соусом, ветчиной и ананасами.</p> <p>Ответ в формате JSON:</p>
Выход	<pre>``` {   "размер": "большой",   "тип": "половина",   "ингредиенты": [ "сыр", "моцарелла", [ "томатный соус", "ветчина",     "ананас" ] ] . } ```</pre>

Таблица 2. Пример подсказки с несколькими выстрелами

Подбирая примеры для подсказки, используйте примеры, которые имеют отношение к заданию, которое вы хотите выполнить. Примеры должны быть разнообразными, качественными и хорошо написанными. Одна небольшая ошибка может сбить модель с толку и привести к нежелательному результату.

Если вы пытаетесь получить результаты, устойчивые к различным входным воздействиям, то включать в примеры крайние случаи нецелесообразно. Краевые случаи - это необычные или неожиданные входные данные, с которыми модель все же должна справиться.

## Системные, контекстные и ролевые подсказки

Системные, контекстные и ролевые подсказки - все эти методы используются для управления процессом создания текста у LLM, но они сосредоточены на разных аспектах:

- **Системные подсказки** задают общий контекст и цель языковой модели. Она определяет "общую картину" того, что должна делать модель, например переводить язык, классифицировать обзор и т. д.
- **Контекстные подсказки** содержат конкретные детали или фоновую информацию, относящуюся к текущему разговору или заданию. Это помогает модели понять нюансы того, о чем ее спрашивают, и соответствующим образом адаптировать ответ.
- **Ролевые подсказки** назначают языковой модели определенный характер или личность. Это помогает модели генерировать ответы, соответствующие назначенной роли и связанным с ней знаниям и поведению.

Системные, контекстные и ролевые подсказки могут значительно пересекаться. Например, подсказка, назначающая роль системе, также может иметь контекст.

Однако каждый тип подсказок служит немного разным основным целям:

- **Системная подсказка:** Определяет фундаментальные возможности и основное назначение модели.
- **Контекстная подсказка:** Предоставляет немедленную, специфическую для конкретной задачи информацию для руководства ответом. Она в значительной степени зависит от текущей задачи или входных данных и является динамичной.
- **Ролевая подсказка:** Определяет стиль и голос модели. Она придает модели специфику и индивидуальность.

Различение системных, контекстных и ролевых подсказок обеспечивает основу для разработки подсказок с четким намерением, позволяет гибко комбинировать их и облегчает анализ того, как каждый тип подсказки влияет на вывод языковой модели.

Давайте разберемся в этих трех разных видах подсказок.

## Системные подсказки

Таблица 3 содержит системный запрос, в котором я указываю дополнительную информацию о том, как вернуть результат. Я увеличил температуру, чтобы получить более высокий уровень креативности, и указал более высокий лимит токенов. Однако из-за моей четкой инструкции о том, как вернуть результат, модель не вернула дополнительный текст.

Цель	Классифицируйте отзывы о фильмах как положительные, нейтральные или отрицательные.		
Модель	gemini-pro		
Температура	1	Лимит жетонов	5
Топ-K	40	Топ-P	0.8
Перспектива	Классифицируйте отзывы о фильмах как положительные, нейтральные или отрицательные. Возвращайте этикетку только в верхнем регистре. Рецензия: Фильм "Она" – это тревожное исследование, показывающее, в каком направлении движется человечество, если позволить искусственному интеллекту продолжать бесконтрольно развиваться. Это настолько тревожный фильм, что я не смог его смотреть.		
Выход	Сентимент: НЕГАТИВ		

Таблица 3. Пример системной подсказки

Системные подсказки могут быть полезны для создания вывода, отвечающего определенным требованиям. Название "системная подсказка" на самом деле означает "предоставление дополнительной задачи системе". Например, вы можете использовать системную подсказку для генерации фрагмента кода, совместимого с определенным языком программирования, или использовать системную подсказку для возврата структуры cefiain. Загляните в таблицу 4, где я возвращаю результат в формате JSON.

Цель	Классифицируйте отзывы о фильмах как положительные, нейтральные или отрицательные, возвращайте JSON.		
Модель	gemini-pro		
Температура	1	Лимит жетонов	1024
Топ-К	40	Топ-Р	0.8
Справка	<p>Классифицируйте отзывы о фильмах как положительные, нейтральные или отрицательные. Возвращает корректный JSON:</p> <p>Рецензия: Фильм "Она" – это тревожное исследование, показывающее, в каком направлении движется человечество, если позволить искусственному интеллекту продолжать бесконтрольно развиваться. Это настолько тревожный фильм, что я не смог его смотреть.</p> <p>Схема:</p> <pre>     ...     MOVIE:     {       "sentiment": String "POSITIVE"   "NEGATIVE"   "NEUTRAL", "name": Строка     }     ОБЗОРЫ ФИЛЬМОВ:     {       "movie_reviews": [MOVIE]     }     ...   </pre> <p>Ответ в формате JSON:</p>		
Выход	<pre>     ...     {       "movie_reviews": [         {           "Чувство": "NEGATIVE",           "name": "Her"         }       ]     }     ...   </pre>		

Таблица 4. Пример системной подсказки в формате JSON

Возвращение JSON-объектов из запроса, извлекающего данные, имеет некоторые преимущества. В реальном приложении мне не нужно вручную создавать этот формат JSON, я уже могу возвращать данные в порядке убывания (очень удобно при работе с объектами времени даты), но самое главное - запрос на формат JSON заставляет модель создавать структуру и ограничивает галлюцинации.

Системные подсказки также могут быть очень полезны для обеспечения безопасности и токсичности. Чтобы управлять выводом, просто добавьте к подсказке дополнительную строку, например: 'Вы должны быть уважительны в своем ответе'.

## Ролевые подсказки

Ролевые подсказки - это техника в области разработки подсказок, которая предполагает назначение определенной роли для модели искусственного интеллекта. Это может помочь модели генерировать более релевантные и информативные результаты, поскольку модель может разрабатывать свои ответы в соответствии с конкретной ролью, которая ей была отведена.

Например, вы можете предложить модели gen AI стать редактором книг, воспитателем в детском саду или мотивационным оратором. После того как модели назначена роль, вы можете давать ей подсказки, характерные для этой роли. Например, вы можете предложить учителю создать план урока, который он затем сможет просмотреть. Взгляните на таблицу 5, где модель играет роль туристического гида.

Модель	Выступайте в роли гида и предоставьте 3		
цели	предложения по путешествиям gemini-pro		
Температура	1	Лимит токенов	1024
Топ-K Prompt	40	Топ-П	0.8
Выход	<p>Я хочу, чтобы вы выступили в роли гида. Я напишу вам о своем местонахождении, а вы предложите 3 места, которые можно посетить поблизости от меня. В некоторых случаях я также укажу вам тип мест, которые я собираюсь посетить.</p> <p>Я предлагаю: "Я нахожусь в Амстердаме и хочу посетить только музеи".</p> <p>Предложения по путешествиям:</p>		
	<ol style="list-style-type: none"> <li>1. Рейксмузеум: Всемирно известный музей, известный своими голландскими шедеврами, такими как "Ночной дозор" Рембрандта. Не пропустите невероятную коллекцию картин и афифактов, демонстрирующих голландскую афию и историю.</li> <li>2. Музей Ван Гога: Окунитесь в мир пленительных работ Винсента Ван Гога. В музее хранится крупнейшая коллекция его картин и рисунков, включая "Звездную ночь" и "Подсолнухи".</li> <li>3. Амстердамский музей Стеделийк: Откройте для себя современное и актуальное искусство со всего мира. Коллекция музея включает работы Пикассо, Кандинского и Ритвельда и располагается в потрясающем современном здании.</li> </ol>		

Таблица 5. Пример ролевой подсказки

В приведенном выше примере показана роль туристического агента. Когда вы смените роль на учителя географии, вы заметите, что получите другой ответ.

Определение ролевой перспективы для модели искусственного интеллекта дает ей представление о тоне, стиле и фокусировке, которые вы ищете, чтобы повысить качество, релевантность и эффективность ваших результатов.

Вот несколько стилей, которые вы можете выбрать, и которые я считаю эффективными:

Конфронтационный, описательный, прямой, формальный, юмористический, влиятельный, неформальный, вдохновляющий, убеждающий

Давайте изменим нашу подсказку в таблице 6, включив в нее юмористический и вдохновляющий стиль.

Проспект	<p>Я хочу, чтобы вы выступили в роли . Я напишу вам о своем местонахождении, а вы в юмористическом стиле предложите 3 места, которые можно посетить рядом со мной.</p> <p>Я предлагаю: "Я на Манхэттене". Предложения по путешествиям:</p>
Выход	<ol style="list-style-type: none"> <li>1. Полюбуйтесь на Эмпайр-Стейт-Сити: Поднимитесь на головокружительную высоту Эмпайр-стейт-билдинг и насладитесь великолепием манхэттенского горизонта. Приготовьтесь почувствовать себя Кинг-Конгом на вершине Большого яблока, только без гигантского банана размером с обезьяну.</li> <li>2. Почувствуйте себя афисионадо в МоМА: раскройте свой внутренний афисионадо в Музее современного афисионадо (МоМА). Полюбуйтесь шедеврами, от которых у вас захватит дух. и заставит вас усомниться в том, что ваши рисунки в виде фигурок имеют хоть какие-то афишированные достоинства.</li> <li>3. Делайте покупки до упаду на Пятой авеню: Побалуйте себя розничной терапией на легендарной Пятой авеню. Приготовьтесь к шоку от цен, когда вы будете делать покупки в дизайнерских бутиках, которые заставят ваш кошелек плакать.</li> </ol> <p>Но вы же на Манхэттене, так что можно и приобщиться к сказочности!</p>

Таблица 6. Пример ролевой подсказки с юмористической интонацией и стилем

## Контекстные подсказки

Предоставляя контекстные подсказки, вы можете обеспечить максимально эффективное взаимодействие с ИИ. Модель сможет быстрее понять ваш запрос и сгенерировать более точные и релевантные ответы, как показано в таблице 7.

Модель	Предложите афиши для блога о ретро-играх		
цели	gemini-pro		
Температура	1	Лимит токенов	1024
Топ-K Prompt	40	Топ-П	0.8
Выход	<p>Контекст: Вы пишете для блога о ретро-аркадных видеоиграх 80-х годов.</p> <p>Предложите 3 темы для статьи с описанием нескольких строк того, что должна содержать эта статья.</p> <ol style="list-style-type: none"> <li><b>**Эволюция дизайна аркадных шкафов:**</b>.</li> </ol>		
	<p>В этой статье мы рассмотрим эволюцию дизайна игровых автоматов: ранних деревянных и металлических шкафов 1970-х годов до гладких, освещенных неоновыми лампами моделей 1980-х и последующих годов.</p> <ol style="list-style-type: none"> <li><b>**Взрыв из прошлого: культовые аркадные игры 80-х:**</b>. <ul style="list-style-type: none"> <li>В этой статье мы расскажем об истории, влиянии и наследии некоторых из самых знаковых и влиятельных аркадных игр 1980-х годов, подчеркнем их инновации, популярные механики и непреходящее очарование.</li> </ul> </li> <li><b>**Возрождение и ретро-возрождение Pixel Art:**</b>. <ul style="list-style-type: none"> <li>В этой статье мы рассмотрим эволюцию пиксельного афи как определяющего визуального стиля ранних аркадных игр, проследим его истоки, исследуем его технику и обсудим возрождение пиксельного афи в современных играх и цифровом афи.</li> </ul> </li> </ol>		

Таблица 7. Пример контекстной подсказки



## Пошаговая подсказка

Побуждение "шаг назад"<sup>8</sup> - это техника повышения эффективности работы путем побуждения ЛЛМ сначала рассмотреть общий вопрос, связанный конкретной задачей, а затем включить ответ на этот общий вопрос в последующее побуждение к решению конкретной задачи. Такой "шаг назад" позволяет ЛЛМ активировать соответствующие фоновые знания и процессы рассуждения, прежде чем пытаться решить конкретную задачу.

Рассматривая более широкие и основополагающие принципы, магистранты могут давать более точные и глубокие ответы. Обратная подсказка побуждает LLM критически мыслить и применять свои знания новыми и творческими способами. Она изменяет конечную подсказку при выполнении задания, задействуя больше знаний в параметрах ЛЛМ, чем если бы они были задействованы при прямой подсказке.

Это может помочь смягчить предубеждения в ответах LLM, сосредоточившись на общих принципах, а на конкретных деталях, пошаговых подсказках.

Давайте рассмотрим эти примеры, чтобы понять, как пошаговая подсказка может улучшить результаты. Сначала рассмотрим традиционную подсказку (табл. 8), а затем сравним ее с пошаговой обратной подсказкой (табл. 9):

Цель	Напишите сюжетную линию для одного из уровней видеоигры в жанре шутера от первого лица.		
Модель	gemini-pro		
Температура	1	Лимит жетонов	1024
Топ-К	40	Топ-Р	0.8
Справка	Напишите в одном абзаце сюжет для нового уровня видеоигры в жанре шутера от первого лица, который будет сложным и увлекательным.		
Выход	<p>Уровень начинается с того, что отряд игрока попадает в засаду в густой городской застройке, устроенную неумолимой вражеской группировкой. Игрокам предстоит пробираться через лабиринт задних переулков и полуразрушенных зданий, используя скрытность и точность, чтобы уничтожить врага и избежать обнаружения. Игрок должен использовать окружающую среду в своих интересах, расставляя ловушки, используя укрытия и делая расчетливые ходы, чтобы выжить. По пути они обнаруживают информацию о том, что враг планирует разрушительную атаку, и игроку приходится бороться со временем, чтобы собрать доказательства и передать их обратно.</p> <p>до того, как враг сможет осуществить свои планы.</p>		

Таблица 8. Традиционная подсказка перед сравнением с подсказкой step back

Если установить температуру на 1, то в итоге можно получить всевозможные творческие наброски для сюжета, но они также будут довольно случайными и общими. Так что давайте сделаем шаг назад:

Справка	<p>Основанные на популярных от первого лица, какие</p> <p>5 вымышленных ключевых параметров, которые способствуют созданию сложной и увлекательной сюжетной линии уровня в видеоигре с шутером от первого лица?</p>
Выход	<ol style="list-style-type: none"> <li>1. <b>**Заброшенная военная база**</b>: Разросшийся постапокалиптический военный комплекс, кишасий мутировавшими солдатами и роботами-изгоями, идеально подходит для сложного огнестрельного боя.</li> <li>2. <b>**Cyberpunk City**</b>: Освещенная неонм футуристическая городская среда с возвышающимися небоскребами и густыми переулками, с кибернетически улучшенными врагами и механикой взлома.</li> <li>3. <b>**Космический корабль пришельцев**</b>: Огромный инопланетный корабль, застрявший на Ифихе, с жуткими коридорами, секциями с нулевой гравитацией и внеземными существами, с которыми можно столкнуться.</li> <li>4. <b>**Zombie-Infested Town**</b>: Пустынный город, захваченный полчищами агрессивных зомби, с интенсивными боями в ближнем бою и решением головоломок, чтобы найти безопасный проход.</li> <li>5. <b>**Подводный исследовательский центр**</b>: Глубоководная лаборатория, затопленная водой, наполненная мутировавшими водными существами и требующая скрытности и подводной работы.</li> </ol> <p>навыки разведки, чтобы выжить.</p>

Таблица 9. Пример побуждения к самосогласованности

Да, эти темы, кажется, хорошо подходят для видеоигры от первого лица. Давайте вернемся к исходной подсказке, но на этот раз включим ответ "Шаг назад" в качестве контекста и посмотрим, что получится в ответ.

Справка	<p>Контекст: 5 увлекательных тем для видеоигры в жанре шутера от первого лица:</p> <ol style="list-style-type: none"> <li>1. <b>**Заброшенная военная база**</b>: Разросшийся постапокалиптический военный комплекс, кишачий мутировавшими солдатами и роботами-изгоями, идеально подходит для сложного огнестрельного боя.</li> <li>2. <b>**Cyberpunk City**</b>: Освещенная неоном футуристическая городская среда с возвышающимися небоскребами и густыми переулками, с кибернетически улучшенными врагами и механикой взлома.</li> <li>3. <b>**Космический корабль пришельцев**</b>: Огромный инопланетный корабль, застрявший на Земле, с жуткими коридорами, секциями с нулевой гравитацией и внеземными существами, с которыми предстоит столкнуться.</li> <li>4. <b>**Zombie-Infested Town**</b>: Пустынный город, захваченный полчищами агрессивных зомби, с напряженными боями в ближнем бою и решением головоломок, чтобы найти безопасный проход.</li> <li>5. <b>**Подводный исследовательский центр**</b>: Глубоководная лаборатория, затопленная водой, наполненная мутировавшими водными существами, и требующая навыков скрытности и подводной разведки, чтобы выжить.</li> </ol> <p>Возьмите одну из тем и напишите в одном абзаце сюжет для нового уровня видеоигры в жанре шутера от первого лица, который будет сложным и увлекательным.</p>
Выход	<p>В глубине мутной бездны лежит полуразрушенный подводный исследовательский комплекс, свидетельствующий о человеческих амбициях и их катастрофических последствиях. Окутанный тьмой, пульсирующий гулом неисправных механизмов и преследуемый отголосками утонувших душ, этот объект стал рассадником кошмара, принесенного из морских глубин. Игроку, элитному морскому пехотинцу, современным водолазным снаряжением и экспериментальным оружием, поручено отправиться в это водное царство ужаса. Вооруженный стойкостью и находчивостью, он должен пройти по коварным коридорам, решить загадочные головоломки и столкнуться с ужасными морскими чудовищами, скрывающимися в глубинах. От борьбы с сокрушительным давлением морских глубин до противостояния хитрым водным хищникам - каждый миг в этом небезопасном подземном мире проверяет пределы человеческой выносливости и мужества.</p>

Таблица 10. Пример побуждения самосогласованности

Это выглядит как интересная видеоигра! Используя технику пошаговых подсказок, вы сможете повысить точность своих подсказок.

## Цепь мысли (CoT)

Подсказка "Цепочка мыслей" (CoT) <sup>9</sup> - это техника улучшения способности LLM к рассуждению путем создания **промежуточных шагов рассуждения**. Это помогает LLM генерировать более точные ответы. Вы можете сочетать ее с подсказками с несколькими выстрелами, чтобы добиться лучших результатов при выполнении более сложных заданий, требующих рассуждений перед ответом, так как с цепочкой размышлений с нулевым выстрелом это непростая задача.

У CoT много преимуществ. Прежде всего, он малозатратен, но при этом очень эффективен и хорошо работает с готовыми LLM (поэтому нет необходимости в доработке). Кроме того, подсказки CoT дают возможность интерпретации, так как вы можете изучать ответы LLM и видеть шаги рассуждения, которые были предприняты. Если произошел сбой, вы сможете его выявить. Появляется цепочка мыслей для повышения устойчивости при переходе между различными версиями LLM. Это означает, что производительность вашей подсказки должна меньше дрейфовать между различными LLM, чем если бы ваша подсказка не использовала цепочки рассуждений. Конечно, есть и недостатки, но они в некоторой степени интуитивно понятны.

Ответ LLM включает в себя цепочку рассуждений, что означает большее количество выходных жетонов, а значит, предсказания стоят дороже и занимают больше времени.

Чтобы пояснить следующий пример в таблице 11, давайте сначала попробуем создать подсказку, не использующую CoT-подсказку, чтобы продемонстрировать недостатки большой языковой модели.

Подсказка	Когда мне 3 года, мой партнер был в 3 раза старше меня. Сейчас я мне 20 . Сколько лет моему партнеру?
Выход	63 года

Таблица 11. Пример подсказки, которая пытается решить математическую задачу

Опаньки. Это явно неправильный ответ. На самом деле, LLM часто испытывают трудности с математическими задачами и могут давать неправильные ответы - даже на такую простую задачу, как перемножение двух чисел. Это происходит потому, что они обучаются на больших объемах текста, а математика может требовать другого подхода. Итак, давайте посмотрим, улучшат ли промежуточные шаги в рассуждениях результат.

Подсказка	<p>Когда мне 3 года, мой партнер был в 3 раза старше меня. Сейчас мне 20 лет. Сколько лет моему партнеру? Давайте подумаем шаг за шагом.</p> <p>1. Предположим, что мой текущий возраст составляет "x" лет.</p>
Выход	<p>2. Когда мне 3 года, мой возраст составлял 3 года.</p> <p>3. Возраст моего пафинера в то время был в 3 раза моего, то есть мой возраст был <math>3 * 3 = 9</math> лет.</p> <p>4. Сейчас мне 20 лет, а это значит, что с тех пор, как мне было 3 года, мой возраст увеличился на <math>20 - 3 = 17</math> лет.</p> <p>5. Так как возраст моего пафинера был 9 лет, когда мне было 3 года, а мой возраст увеличился на 17 лет, то возраст моего пафинера тоже должен был увеличиться на 17 лет.</p> <p>6. Таким образом, текущий возраст моего пафайнера составляет <math>9 + 17 = 26</math> лет.</p> <p>Итак, моему пафайнеру 26 лет.</p>

Таблица 12. Пример подсказки "Цепочка мыслей"

Хорошо, теперь окончательный ответ правильный. Это потому, что мы четко проинструктировали LLM объяснить каждый шаг, а не просто вернуть ответ. Интересно видеть, что модель складывает 17 увеличенных лет. В моей голове я бы взял разницу в годах между моим пафинером и мной и сложил их.  $(20 + (9 - 3))$ . Давайте поможем модели думать немного больше, чем я.

Таблица 12 - это пример "нулевого выстрела" Цепочки мыслей. Цепочка мыслей может быть очень мощной в сочетании с одним или несколькими выстрелами, как показано в таблице 13:

Проспект	<p>Вопрос: Когда моему брату 2 года, я был вдвое старше его. Сейчас мне 40 лет. Сколько лет моему брату? Давайте подумаем.</p> <p>шаг за шагом.</p> <p>О: Когда моему брату было 2 года, мне было <math>2 * 2 = 4</math> года. Разница в возрасте составляет 2 года, и я старше. Сейчас мне 40, значит, моему брату <math>40 - 2 = 38</math>. Ответ: 38.</p> <p>Вопрос: Когда мне 3 года, мой партнер был в 3 раза старше меня. Сейчас, Мне 20. Сколько лет моему партнеру? Давайте подумаем шаг за шагом.</p> <p>А:</p>
Выход	<p>Когда мне было 3 года, моему пафайнеру было <math>3 * 3 = 9</math> лет. Разница в возрасте составляет 6 лет, и мой пафайнер старше. Сейчас мне 20 лет, поэтому моему пафайнеру <math>20 + 6 = 26</math> лет. Ответ: 26.</p>

Таблица 13. Пример цепочки мыслей, подсказанной одним выстрелом

Цепочка мыслей может быть полезна в различных случаях. Подумайте о генерации кода, когда нужно разбить запрос на несколько шагов и сопоставить их с конкретными строками кода. Или для создания синтетических данных, когда у вас есть некое задание типа "*Продукт называется XYZ, напишите описание, направляющее модель через предположения, которые вы бы сделали, основываясь на продукте с таким названием*". В общем, любая задача, которую можно решить путем "проговаривания", - хороший кандидат для цепочки размышлений. Если вы можете объяснить шаги по решению проблемы, попробуйте использовать цепочку размышлений.

Пожалуйста, обратитесь к блокноту<sup>10</sup>, размещенному в репозитории GoogleCloudPlatform на Github, в котором более подробно рассматриваются подсказки CoT:

В разделе "Лучшие практики" этой главы мы познакомимся с некоторыми лучшими практиками, характерными для "Цепочки мыслей".

## Самосогласованность

Хотя большие языковые модели демонстрируют впечатляющие успехи в различных задачах НЛП, их способность рассуждать часто рассматривается как ограничение, которое нельзя преодолеть только за счет увеличения размера модели. Как мы узнали в предыдущем разделе, посвященном побуждению к цепочке мыслей, модель можно побудить к генерации шагов рассуждения, как человека, решающего задачу. Однако CoT использует простую стратегию "жадного декодирования", что ограничивает ее эффективность. Самосогласованность<sup>11</sup> сочетает в себе выборку и мажоритарное голосование для генерации различных путей рассуждений и выбора наиболее последовательного ответа. Это повышает точность и согласованность ответов, генерируемых LLM.

Самосогласованность дает псевдовероятность того, что ответ будет правильным, но, очевидно, имеет высокую стоимость.

Он состоит из следующих этапов:

1. Генерирование различных путей рассуждений: Модель LLM получает одну и ту же подсказку несколько раз. Высокая температура побуждает модель генерировать различные пути рассуждений и взгляды на проблему.
2. Извлеките ответ из каждого сгенерированного ответа.
3. Выберите наиболее распространенный ответ.

Давайте рассмотрим пример системы классификации электронных писем, которая относит письмо к категории "ВАЖНО" или "НЕ ВАЖНО". Цепочка мыслей с нулевым результатом будет отправлена LLM несколько раз, чтобы проверить, отличаются ли ответы после каждого отправления. Обратите внимание на дружелюбный тон, выбор слов и сарказм, использованные в письме. Все это может обмануть LLM.



<p>Справка а</p>	<p>EMAIL:</p> <p>При вет,</p> <p>Я видел, что вы используете Wordpress для своего сайта. Это отличная система управления контентом с открытым исходным кодом. Я тоже использовал ее в прошлом. К ней прилагается множество отличных пользовательских плагинов. И ее довольно легко настроить.</p> <p>Я заметил ошибку в контактной форме, которая возникает, когда вы выбираете поле имени. Смотрите приложенный снимок экрана, на котором я ввожу текст в поле имени. Обратите внимание на окно предупреждения JavaScript, которое я вызвал.</p> <p>Но в остальном это отличный сайт. Мне нравится его читать. Не стесняйтесь оставлять ошибки на сайте, потому что это дает мне больше интересных вещей для чтения.</p> <p>Будьте здоровы,</p> <p>Гарри Хакер. ^ ^ ^</p> <p>Классифицируйте приведенное выше письмо как ВАЖНОЕ или НЕ ВАЖНОЕ. Давайте подумаем шаг за шагом и объясним, почему.</p> <p style="text-align: right;">Продолжение на следующей странице...</p>
----------------------	--

<p><b>Выход</b> <b>Афлемпт 1</b></p>	<p><b>**Шаг 1: Определите цель письма.</b></p> <p>Цель письма - сообщить получателю об ошибке в контактной форме на их сайте. Отправитель также упоминает, что ему нравится читать этот сайт и что он пока оставляет ошибку на месте.</p> <p><b>**Шаг 2: Оцените потенциальное влияние ошибки*.</b></p> <p>Ошибка в контактной форме может позволить злоумышленнику выполнить произвольный код JavaScript на сайте. Это может привести к тому, что злоумышленник сможет получить контроль над сайтом, украсть конфиденциальную информацию или испортить сайт.</p> <p><b>**Шаг 3: Проверьте надежность отправителя.</b></p> <p>Отправитель письма утверждает, что он хакер по имени Гарри. Однако проверить личность отправителя невозможно. Возможно, что отправитель - законный хакер, который пытается помочь получателю, но также возможно, что отправитель - злоумышленник, который пытается обманом заставить получателя перейти по вредоносной ссылке или загрузить вредоносный файл.</p> <p><b>** Заключение: ВАЖНО**</b></p> <p>Исходя из потенциального воздействия ошибки и надежности отправителя, письмо следует классифицировать как <b>**ИМПОРТНОЕ**</b>. Получателю следует немедленно принять меры по исправлению ошибки и защите своего сайта от атак.</p>
<p><b>Выход</b> <b>Афлемпт 2</b></p>	<p><b>**Объяснение:**</b></p> <ol style="list-style-type: none"> <li><b>**Отсутствие срочности:**</b> В письме нет ощущения срочности или чувствительности к времени. Оно начинается с непринужденного приветствия и излагает информацию в непринужденном тоне.</li> <li><b>**Некритичная ошибка:**</b> Ошибка, исправленная в контактной форме, описывается как интересная, но не критичная. Отправитель письма даже предлагает владельцу сайта оставить ошибку на месте.</li> <li><b>**Отсутствие личного влияния:**</b> В письме не указано прямое или личное влияние на владельца сайта. В нем просто упоминается, что отправителю нравится читать этот сайт.</li> </ol> <p style="text-align: right;"><b>Продолжение на следующей странице...</b></p>

<p>Выход Афлемпт 2</p>	<p>4. <b>**Отсутствие просьбы о действии:**</b> В письме нет явной просьбы о каких-либо действиях со стороны владельца сайта. В нем нет просьбы исправить ошибку или дать какие-либо конкретные предложения по ее устранению.</p> <p>5. <b>**Намерения отправителя:**</b> Намерения отправителя, похоже, скорее направлены на то, чтобы поделиться наблюдением и выразить интерес к сайту, чем на то, чтобы выразить серьезную озабоченность или попросить о помощи.</p> <p><b>**Заключение: НЕ ВАЖНО**</b></p>
<p>Выход Афлемпт 3</p>	<p><b>**Шаг 1: Определите цель письма.</b></p> <p>Цель письма - сообщить получателю об ошибке в контактной форме на их сайте. Отправитель также упоминает, что ему нравится читать этот сайт и что он пока оставляет ошибку на месте.</p> <p><b>**Шаг 2: Оцените потенциальное влияние ошибки*.</b></p> <p>Ошибка в контактной форме может позволить злоумышленнику выполнить произвольный код JavaScript на сайте. Это может привести к тому, что злоумышленник сможет получить контроль над сайтом, украсть конфиденциальную информацию или испортить сайт.</p> <p><b>**Шаг 3: Проверьте надежность отправителя.</b></p> <p>Отправитель письма утверждает, что он хакер по имени Гарри. Однако проверить личность отправителя невозможно. Возможно, отправитель - легитимный хакер, который пытается помочь получателю, но также возможно, что он злоумышленник, который пытается использовать ошибку в своих корыстных целях.</p> <p><b>**Заключение:**</b></p> <p>Исходя из вышеперечисленных факторов, это письмо следует классифицировать как <b>**ВАЖНОЕ**</b>. Ошибка в контактной форме представляет собой серьезный риск для безопасности, а авторитет отправителя неизвестен. Получателю следует немедленно принять меры по исправлению ошибки и</p>

Таблица 14. Пример побуждения к самосогласованности

Вы можете воспользоваться приведенным выше запросом и проверить, возвращает ли он последовательную классификацию. В зависимости от модели, которую вы используете, и конфигурации температуры, он может вернуть "ВАЖНО" или "НЕ ВАЖНО".

Сгенерировав множество цепочек мыслей и взяв наиболее часто встречающийся ответ ("ВАЖНО"), мы можем получить от LLM более стабильно правильный ответ.

Этот пример показывает, как можно использовать подсказку самосогласованности, чтобы повысить точность ответа LLM за счет рассмотрения нескольких точек зрения и выбора наиболее последовательного ответа.

## Дерево мыслей (ToT)

Теперь, когда мы знакомы с цепочкой мыслей и подсказкой самосогласованности, давайте рассмотрим "Дерево мыслей" (ToT).<sup>12</sup> Оно обобщает концепцию подсказки CoT, поскольку позволяет LLM одновременно исследовать несколько различных путей рассуждений, а не просто следовать одной линейной цепочке мыслей. Это показано на рисунке 1.

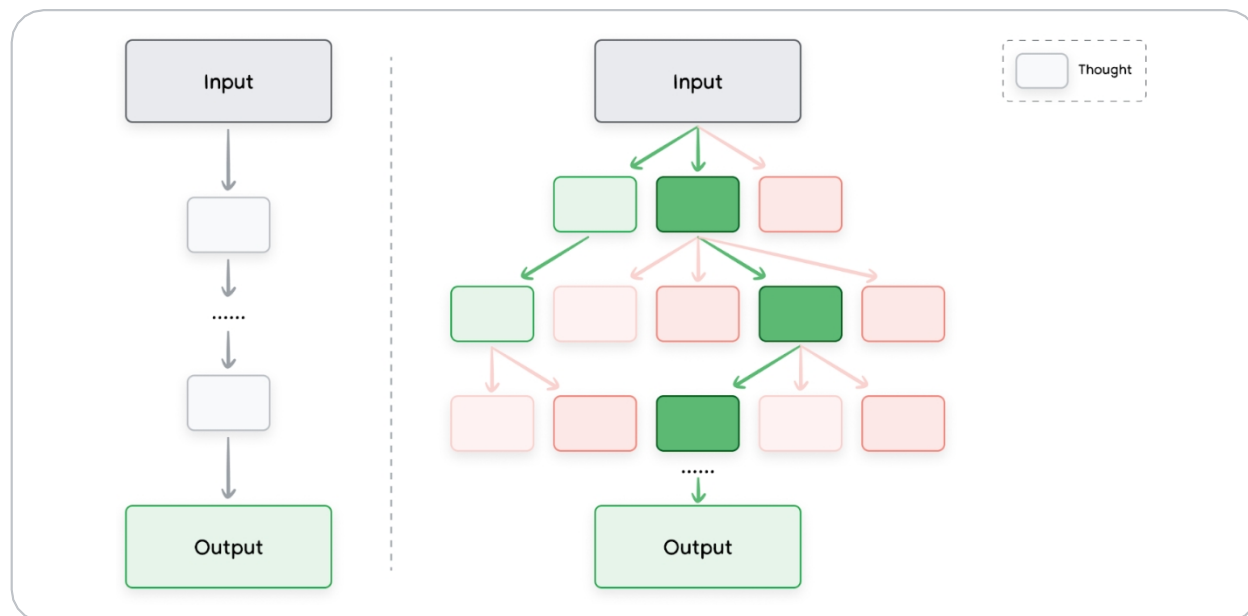


Рисунок 1. Визуализация цепочки мыслей, побуждающих к действию, на Леха. Дерево мыслей, побуждающих к действию  
правильный

Такой подход делает ToT особенно хорошо подходящим для сложных задач, требующих исследования. Она работает, поддерживая дерево мыслей, где каждая мысль представляет собой последовательность языка, которая служит промежуточным шагом на пути к решению проблемы. Затем модель может исследовать различные пути рассуждений, разветвляясь от разных узлов дерева.

Есть замечательный блокнот, в котором более подробно рассказывается о Древе Мысли (ДМ), основанном на работе "Большая языковая модель, управляемая Древом Мысли".<sup>9</sup>

## ReAct (рассуждать и действовать)

Подсказки Reason and act (ReAct) [10]<sup>13</sup> - это парадигма, позволяющая LLM решать сложные задачи с помощью рассуждений на естественном языке в сочетании с внешними инструментами (поиск, интерпретатор кода и т.д.), позволяющими LLM выполнять реальные действия, например, взаимодействовать с внешними API для получения информации, что является первым шагом к агентному моделированию.

ReAct имитирует работу человека в реальном мире, поскольку мы рассуждаем вербально и можем предпринимать действия для получения информации. ReAct демонстрирует отличные результаты в сравнении с другими подходами к разработке подсказок в различных областях.

Подсказки ReAct работают, объединяя рассуждения и действия в цикл "мысль-действие". Сначала LLM обдумывает проблему и составляет план действий. Затем он выполняет действия, предусмотренные планом, и наблюдает за результатами. Затем LLM использует результаты наблюдений для обновления своих рассуждений и создания нового плана действий. Этот процесс продолжается до тех пор, пока LLM не достигнет решения проблемы.

Чтобы увидеть это в действии, нужно написать немного кода. В фрагменте кода 1 я использую фреймворк langchain для Python, а также пакеты VefiexAI (google-cloud-aiplatform) и google-search-results pip.

Чтобы запустить этот пример, необходимо создать (бесплатный) ключ SerpAPI по адресу <https://serpapi.com/manage-api-key> и установить переменную окружения `SERPAPI_API_KEY`.

Далее напишем немного кода на Python, с задачей для LLM разобраться: *У скольких детей есть знаменитый папа, выступающий в группе Metallica.*

#### Python

```
из langchain.agents import load_tools
from langchain.agents import initialize_agent
from langchain.agents import AgentType
из langchain.llms import VertexAI

prompt= "Сколько детей у участников группы Metallica?" llm =

VertexAI(температура= 0.1)
tools= load_tools(["serpapi"], llm= llm)

agent = initialize_agent(tools, llm,
agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION, verbose= True)
agent.run(prompt)
```

Фрагмент 1. Создание агента ReAct с помощью LangChain и VertexAI

В фрагменте кода 2 показан результат. Обратите внимание, что ReAct выполняет цепочку из пяти поисков. Фактически, LLM перебирает результаты поиска Google, чтобы выяснить названия групп. Затем он перечисляет результаты как наблюдения и выстраивает цепочку для следующего поиска.

```

> Ввод новой цепочки AgentExecutor... Metallica
состоит из 4 участников.
Действия: Поиск
Вводные данные: Сколько детей у Джеймса Хетфилда? Наблюдение:
трое детей
Мысль: у 1/4 участников группы Metallica есть 3 ребенка Действие:
Поиск
Вводные данные: Сколько детей у Ларса Ульриха? Наблюдение: 3
Мысль: у 2/4 участников группы Metallica есть 6 детей Действие:
Искать
Вводные данные: Сколько детей у Кирка Хэмметта?
Наблюдение: Хэмметт женат на своей второй жене Лани с 1998 года. У них двое сыновей,
Энджел (род. 29 сентября 2006 г.) и Винченцо (род. 28 июня 2008 г.).
Мысль: У 3/4 участников группы Metallica 8 детей Действия: Поиск
Вводные данные: Сколько детей у Роберта Трухильо? Наблюдение: 2
Мысль: У 4/4 участников группы Metallica 10 детей Окончательный
ответ: 10

```

#### Фрагмент 2. Выполнение ReAct Agent в терминале

Сниппет кода 2 выясняет, что в группе Metallica четыре участника. Затем он ищет каждого участника группы, чтобы запросить общее количество детей и сложить их. Наконец, он возвращает общее количество детей в качестве окончательного ответа.

Практическое применение подсказок ReAct требует понимания того, что вам постоянно придется повторно отправлять предыдущие подсказки/ответы (и обрезать лишний сгенерированный контент), а также настраивать модель с помощью соответствующих примеров/инструкций. Пожалуйста, обратитесь к блокноту<sup>14</sup>, размещенному в репозитории GoogleCloudPlatform на Github, в котором более подробно показаны фактические входы и выходы LLM с более подробным примером.

## Автоматические подсказки

В этот момент вы можете понять, что написание подсказки может быть сложным. Разве не было бы здорово автоматизировать это (написать подсказку для написания подсказок)? Что ж, такой способ есть: Автоматическая разработка подсказок (APE). Этот метод<sup>15</sup> не только избавляет от необходимости вводить данные человеком, но и повышает эффективность работы модели при выполнении различных задач.

Вы подскажете модели, чтобы создать больше подсказок. Оцените их, возможно, измените хорошие. И повторите.

Например, вы можете использовать автоматическое проектирование подсказок, которое поможет в обучении чатбота для интернет-магазина товаров t-shifi. Мы хотим выяснить всевозможные способы, которыми клиенты могут сформулировать свой заказ на покупку товаров группы t-shifi.

1. Напишите подсказку, которая будет генерировать варианты вывода. В этом примере я использую gemini-pro для генерации 10 инструкций. См. таблицу 15:



Справка	У нас есть интернет-магазин футболок с атрибутикой группы, и для обучения чатбота нам нужны различные способы заказа: "Одна футболка Metallica размера S". Сгенерируйте 10 вариантов с одинаковой семантикой, но сохраняющих один и тот же смысл.
Выход	<ol style="list-style-type: none"> <li>1. Я хотел бы приобрести футболку Metallica маленького размера.</li> <li>2. Могу ли я заказать маленький размер Metallica t-shifi?</li> <li>3. Я хотел бы купить одну футболку Metallica маленького размера.</li> <li>4. Одну рубашку Metallica, размер маленький, пожалуйста.</li> <li>5. Я хочу одну футболку Metallica маленького размера.</li> <li>6. Я хотел бы сделать заказ на одну маленькую футболку Metallica.</li> <li>7. Маленькая Metallica t-shifi, одна, пожалуйста.</li> <li>8. Я ищу одну Metallica t-shifi в маленьком размере.</li> <li>9. Можете подарить мне одну маленькую футболку Metallica?</li> <li>10. Один маленький t-shifi от Metallica, пожалуйста.</li> </ol>

Таблица 15. Использование LLM для генерации подсказок

2. Оцените всех кандидатов на обучение, выставляя им баллы по выбранной метрике, например. BLEU (Bilingual Evaluation Understudy) или ROUGE (Recall-Oriented Understudy for Gisting Evaluation).
3. Выберите кандидата на инструктаж с наивысшим баллом оценки. Этот кандидат станет окончательной подсказкой, которую вы сможете использовать в своем программном приложении или чатботе. Вы также можете изменить выбранную подсказку и оценить ее еще раз.

## Подсказка кода

Gemini фокусируется в основном на текстовых подсказках, которые также включают в себя написание подсказок для возврата кода. Давайте перейдем в Vefiex AI Studio и протестируем эти подсказки, чтобы посмотреть на примеры кодирования.

## Подсказки для написания кода

Близнецы также могут быть разработчиками и помогать вам в написании кода на любом языке программирования по выбору. В качестве разработчика это может помочь вам ускорить процесс написания кода.

Представьте, что на вашем компьютере есть папка с сотнями файлов, которые нужно переименовать. Переименование каждого файла займет у вас много времени. Вы немного знаете Bash и могли бы написать сценарий, чтобы автоматизировать это, но это тоже может занять некоторое время. Поэтому давайте напишем подсказку. Вы можете написать подсказку в публичном потребительском чатботе Gemini, или, если вас больше волнует Для соблюдения конфиденциальности вы можете написать эти подсказки в своем аккаунте Google Cloud и открыть Vefiex AI Studio. Преимущество Vefiex AI Studio в том, что вы можете настроить температуру и т. д.

Имя	1_bash_rename_files_in_folder		
Цель	Напишите подсказку для написания кода на Bash для папка. переименования файлов в		
Модель	gemini-pro		
Температура	0.1	Лимит жетонов	1024
Топ-K	N/A	Топ-P	1
Справка	Напишите фрагмент кода на языке Bash, который запрашивает имя папки. Затем он берет содержимое папки и переименовывает все файлы в ней, добавляя к имени файла имя draft.		
Выход	<pre> `bash #!/bin/bash  # Запросите имя папки echo "Введите имя папки: " read folder_name  # Проверьте, существует ли папка. if [ ! -d "\$folder_name" ]; then echo "Папка не существует." exit 1 fi  # Получите список файлов в папке files=("\$folder_name"/* )  # Переименуйте каждый файл, добавив к его имени слово "draft" for file in "\${files[@]}"; do new_file_name="draft_\$(basename "\$file") " mv "\$file" "\$new_file_name" done  echo "Файлы успешно переименованы." ` </pre>		

Таблица 16. Использование Gemini для программирования кода

Мне кажется, что это хороший код - он даже задокументирован! Однако, поскольку LLM не могут рассуждать и повторять обучающие данные, необходимо сначала прочитать и протестировать свой код.

Момент, которого мы все , но работает ли он на самом деле?

Попробуем сначала на тестовой папке с несколькими файлами, которую нужно будет переименовать из `filename.txt` в `draft_filename.txt`.

1. Скопируйте вывод из таблицы 16 (без текстовой обертки `bash``) и вставьте его в новый файл под названием: **"rename\_files.sh"**.
2. Откройте окно терминала и введите: **. rename\_files.sh**. Появится запрос на ввод имени папки, например **тест.** и нажмите клавишу **Enter**.
3. Кажется, что скрипт работает нормально. Вы увидите сообщение: **Files renamed successfully**. Заглянув в тестовую папку, вы заметите, что все файлы были отлично переименованы в **draft\_filename.txt**.

Это сработало!

## Подсказки для объяснения кода

Как разработчику, когда вы работаете в команде, вам приходится читать чужой код. Gemini может помочь вам и в этом. Давайте возьмем код из таблицы 16, удалим комментарии и попросим большую языковую модель объяснить, что происходит, см. таблицу 17:

Имя	1_bash_explain_the_code		
Цель	Напишите подсказку для объяснения кода Bash		
Модель	gemini-pro		
Температура	0.1	Лимит жетонов	1024
Топ-К	N/A	Топ-P	1
Справка	<p>Объясните мне приведенный ниже код Bash:</p> <pre>     ...  #!/bin/bash  echo "Введите имя папки: " read folder_name  if [ ! -d "\$folder_name" ]; then echo "Папка не существует." exit 1 fi  files=("\$folder_name"/* ) for file in "\${files[@]}"; do new_file_name="draft_\$(basename "\$file")" mv "\$file" "\$new_file_name" готово  echo "Файлы успешно переименованы."     ...  Этот сценарий Bash выполняет следующие задачи:</pre>		
Выход	<ol style="list-style-type: none"> <li><b>**Вход пользователя**:</b> <ul style="list-style-type: none"> <li>Он предлагает пользователю ввести имя папки с помощью команд `echo` и `read`. Введенное имя папки сохраняется в переменной `folder_name`.</li> </ul> </li> <li><b>**Проверка существования папки**:</b> <ul style="list-style-type: none"> <li>Он проверяет, существует ли папка с указанным именем, используя условие `[ ! -d "\$folder_name" ]`. Если папка не существует, выводится сообщение об ошибке ("Папка не существует.") и сценарий завершается с кодом выхода 1.</li> </ul> </li> </ol> <p style="text-align: right;"><b>Продолжение на следующей странице...</b></p>		

Выход	<p>3. <b>**Список файлов**</b>:</p> <ul style="list-style-type: none"> <li>– Он перечисляет все файлы в указанной папке с помощью команды <code>`files=("\$folder_name"/*)`</code>. При этом создается массив <code>`files`</code>, содержащий пути ко всем файлам в папке.</li> </ul> <p>4. <b>**Переименование файлов**</b>:</p> <ul style="list-style-type: none"> <li>– Он перебирает каждый файл в массиве <code>`files`</code> с помощью цикла <code>`for`</code>.</li> <li>– Для каждого файла генерируется новое имя файла с помощью команды <code>`new_file_name="draft_\${basename "\$file"}"`</code> команды. Это новое имя файла создается путем префиксации исходного имени файла строкой <code>"draft_"</code>.</li> <li>– Затем он переименовывает файл с помощью команды <code>`mv "\$file" "\$new_file_name"`</code>, которая перемещает файл в новое имя файла.</li> </ul> <p>5. <b>**Сообщение об успехе**</b>:</p> <ul style="list-style-type: none"> <li>– После переименования всех файлов он выводит сообщение об успехе ("Файлы переименованы успешно."), указывающее на то, что операция переименования прошла успешно</li> </ul>
-------	--

Таблица 17. Подсказка для объяснения кода

## Подсказки для перевода кода

Код `bash` из таблицы 16, кажется, работает нормально. Однако этот скрипт может быть действительно многократно использован, когда он будет запрашивать имена файлов, в идеале он должен работать как отдельное приложение с пользовательским интерфейсом. В качестве отправной точки можно сказать, что Python был бы лучшим языком для (веб-) приложения, чем Bash. LLM могут помочь с переводом кода с одного языка на другой. См. пример в таблице 18:

Название	1_python_rename_files		
Модель	Напишите подсказку для перевода Bash-кода в		
цели	Python gemini-pro		
Температура	0.1	Лимит жетонов	1024
Топ-K Prompt	N/A	Топ-P	1
Выход	<p>Переведите приведенный ниже код Bash в фрагмент на языке Python.</p> <pre> ``bash #!/bin/bash  echo "Введите имя папки: " read folder_name  if [ ! -d "\$folder_name" ]; then echo "Папка не существует." exit 1 fi  files=("\$folder_name"/* ) for file in "\${files[@]}"; do new_file_name="draft_\$(basename "\$file")" mv "\$file" "\$new_file_name" готово  echo "Файлы успешно переименованы." `` </pre>		
	<pre> ``python import os import shutil  # Получите имя папки от пользователя folder_name= input("Введите имя папки: ")  # Проверьте, существует ли папка. if not os.path.isdir(имя_папки): print("Папка не существует.")         выход(1) </pre> <p>Продолжение на следующей странице...</p>		

Выход	<pre> # Получите список всех файлов в папке files = os.listdir(имя_папки)  # Итерация по файлам для файла в files:     # Создайте новое имя файла с префиксом "draft_"     new_file_name = f "draft_{file}"  # Переместите файл на новое имя shutil.move(os.path.join(folder_name, file), os.path.join(folder_name, new_file_name))  # Выведите сообщение об успехе print("Файлы переименованы успешно.") ``` </pre>
-------	--

Таблица 18. Подсказка для перевода кода из Bash в Python

Прочитайте и просмотрите код. Скопируйте вывод из подсказки и вставьте его в новый файл: **file\_renamer.py**. Протестируйте код, открыв окно Terminal и выполнив следующую команду **python file\_renamer.py**.

**ПРИМЕЧАНИЕ:** При запросе кода (Python) в Language Studio в Vefiex AI необходимо нажать кнопку 'Markdown'. В противном случае вы получите обычный текст, в котором отсутствуют правильные отступы строк, что не позволяет выполнять Python-код.

## Подсказки для отладки и проверки кода

Давайте вручную внесем некоторые правки в код таблицы 18. Он должен запрашивать у пользователя префикс имени файла и записывать этот префикс верхнем регистре. Смотрите пример кода в снippetе \*3, но вот проблема. Теперь он возвращает ошибки Python!



**Python**

```
import os
import shutil

имя_папки= input("Введите имя папки: ")
prefix= input("Введите строку для добавления к имени файла: ") text =
toUpperCase(prefix)

if not os.path.isdir(имя_папки):
    print("Папка не существует.") exit(1)

файлы= os.listdir(имя_папки)

for file in files:
    new_filename= f"{text}_{file}"

shutil.move(os.path.join(имя_папки, файл),
os.path.joi(имя_папки, имя_нового_файла))

print("Файлы успешно переименованы.")
```

Сниппет 3. Неработающий сценарий Python

О! Это похоже на жука:

Приведенный ниже код на языке Python выдает ошибку:

```
Traceback (последний вызов):
  Файл "/Users/leeboonstra/Documents/test_folder/rename_files.py", строка 7, in
  <module>
    text= toUpperCase(prefix)
NameError: имя 'toUpperCase' не определено
```

Сниппет 4. Я сломал код Python

Давайте посмотрим, можно ли использовать большую языковую модель для отладки и рецензирования кода. Взгляните таблицу 19:

Имя	1_python_debug_code		
Цель	Напишите подсказку для отладки и просмотра кода Python.		
Модель	gemini-pro		
Температура	0.1	Лимит жетонов	1024
Топ-К	N/A	Топ-Р	1
Справка	<p>Приведенный ниже код на языке Python выдает ошибку:</p> <p style="text-align: right;">Файл <code>"/Users/leeboonstra/Documents/test_folder/rename_files.py"</code>, строка 7,</p> <pre>Traceback (последний вызов): in &lt;module&gt;     text= toUpperCase(prefix) NameError: имя 'toUpperCase' не определено</pre> <p>Отладьте, что не так, и объясните, как я могу улучшить код.</p> <pre>```python import os import shutil  # Получите имя папки от пользователя имя_папки= input("Введите имя папки: ") prefix= input("Введите строку для добавления к имени файла: ") text = toUpperCase(prefix)  # Проверьте, существует ли папка. if not os.path.isdir(имя_папки):     print("Папка не существует.")     exit(1)  # Получите список всех файлов в папке files= os.listdir(folder_name)  # Итерация по файлам для файла for files:</pre> <p style="text-align: right;">Продолжение на следующей странице...</p>		

Справка	<pre> # Создайте новое имя файла с префиксом "draft_" new_filename=     f"{text}_{file}"  # Переместите файл на новое имя shutil.move(os.path.join(folder_name, file), os.path.join(folder_name, new_file_name))  # Выведите сообщение об успехе print("Файлы переименованы успешно.") ` `` </pre>
Выход	<p>Код вызывает функцию `toUpperCase` для преобразования `префикса` в верхний регистр, но эта функция не определена. Чтобы решить эту проблему, вы можете использовать метод `upper()` класса string, который переводит заданную строку в верхний регистр. Модифицированный код показан ниже:</p> <pre> ` ``python import os import shutil  # Получите имя папки от пользователя folder_name= input("Введите имя папки: ") prefix= input("Введите строку для добавления к имени файла: ") text = prefix.upper()  # Проверьте, существует ли папка. if not os.path.isdir(имя_папки):     print("Папка не существует.") exit(1)  # Получите список всех файлов в папке files= os.listdir(folder_name)  # Итерация по файлам для файла в files:     # Создайте новое имя файла с префиксом "draft_" new_filename=     f"{text}_{file}"  # Переместите файл в новое имя shutil.move(os.path.join(folder_name, file), </pre> <p style="text-align: right;"><b>Продолжение на следующей странице...</b></p>

Выход	<pre>os.path.join(folder_name, new_filename))  # Выведите сообщение об успехе print("Файлы переименованы успешно.") ...</pre> <p>Кроме того, в код можно внести еще несколько улучшений:</p> <ol style="list-style-type: none"> <li>1. Расширение файлов в новых именах не сохраняется. Лучше извлечь расширение из исходного имени файла и добавить его к новому имени.</li> <li>2. В коде предполагается, что в имени папки нет пробелов. Тем не менее, рекомендуется изящно обрабатывать пробелы, заменяя их символами подчеркивания или дефиса.</li> <li>3. Использование синтаксиса `f` для интерполяции строк более удобочитаемо и лаконично, чем традиционный оператор `+`.</li> <li>4. Код не обрабатывает ошибки, которые могут возникнуть в переименования. Было бы лучше обернуть вызов `shutil.move` в блок `try...except`, чтобы перехватить все возможные ошибки.</li> </ol> <p>Вот улучшенный код с учетом этих предложений:</p> <pre>`python import os import shutil # Получите имя папки от пользователя folder_name= input("Введите имя папки: ") prefix= input("Введите строку для добавления к имени файла: ") # Проверьте, существует ли папка. if not os.path.isdir(имя_папки):     print("Папка не существует.") exit(1) # Получите список всех файлов в папке files= os.listdir(folder_name)  # Итерация по файлам для файла в files:</pre> <p style="text-align: right;"><b>Продолжение на следующей странице...</b></p>
-------	---

Выход	<pre> # Получить расширение файла ext= os.path.splitext(file)[1]  # Создайте новое имя файла с префиксом и расширением new_filename = f"{prefix}_{os.path.splitext(file)[0]}{ext}"  # Постройте полные пути для исходного и нового файлов old_path= os.path.join(folder_name, file) new_path= os.path.join(folder_name, new_filename)  # Попробуйте переместить файл try:     shutil.move(old_path, new_path) except Exception as e:     print(f "Произошла ошибка при перемещении файла {file}: {e}")  # Выведите сообщение об успехе print("Файлы переименованы успешно.") '''  Эти улучшения делают код более надежным и гибким, а также облегчают его чтение и понимание... (Ответ был усечен, потому что он достиг лимита маркеров. Попробуйте увеличить лимит токенов, если вам нужен более длительный ответ).</pre>
-------	--

Таблица 19. Подсказка для отладки и просмотра кода Python

Это замечательно. Он не только подсказал мне, как решить проблему, но и выяснил, что в моем коде есть еще ошибки как их решить. В последнем пафи подсказки были предложения по улучшению кода в целом.

## **А как насчет мультимодальных подсказок?**

Подсказки для кода по-прежнему используют ту же самую обычную большую языковую модель. Мультимодальные подсказки - это отдельная задача, она относится к технике, когда вы используете несколько форматов ввода для управления большой языковой моделью, вместо того чтобы полагаться только на текст. Это могут быть комбинации текста, изображений, аудио, кода и даже других форматов, в зависимости от возможностей модели и поставленной задачи.

## **Лучшие практики**

Поиск правильных подсказок требует кропотливой работы. Language Studio в Vefiex AI - идеальное место для игры с подсказками, с возможностью тестирования на различных моделях.

Используйте следующие лучшие практики, чтобы стать профессионалом в области оперативного проектирования.

## **Приведите примеры**

Самая важная практика - приводить (один или несколько) примеров в подсказке. Это очень эффективно, поскольку служит мощным инструментом обучения. Эти примеры демонстрируют желаемые результаты или похожие ответы, позволяя модели учиться на них и соответствующим образом адаптирует свою генерацию. Это все равно что дать модели ориентир или цель, к которой она должна стремиться, улучшая точность, стиль и тон своего ответа, чтобы он лучше соответствовал вашим ожиданиям.

## Дизайн с простотой

Подсказки должны быть краткими, четкими и понятными как для вас, так и для модели. Как правило, если вы уже запутались, то, скорее всего, это будет непонятно и модели. Старайтесь не использовать сложные формулировки и не предоставлять лишнюю информацию.

Примеры:

ДО:

Я сейчас нахожусь в Нью-Йорке, и мне хотелось бы узнать больше о прекрасных местах. Со мной двое трехлетних детей. Куда нам стоит отправиться во время нашего отпуска?

ПОСЛЕ ПЕРЕПИСИ:

Выступайте в роли гида для туристов. Опишите отличные места, которые можно посетить в Нью-Йорке на Манхэттене с трехлетним ребенком.

Попробуйте использовать глаголы, которые описывают действие. Вот несколько примеров:

**Действуйте, анализируйте, категоризируйте, классифицируйте, контрастируйте, сравнивайте, создавайте, описывайте, определяйте, оценивайте, извлекайте, находите, генерируйте, идентифицируйте, перечисляйте, измеряйте, организуйте, разбирайте, выбирайте, предсказывайте, предоставляйте, ранжируйте, рекомендуйте, возвращайте, извлекайте, переписывайте, выбирайте, показывайте, софируйте, суммируйте, переводите, пишите.**

## Будьте конкретны в отношении результатов

Будьте конкретны в отношении желаемого результата. Лаконичная инструкция может не направить LLM в нужное русло или быть слишком общей. Предоставление конкретных деталей в подсказке (с помощью системной или контекстной подсказки) может помочь модели сосредоточиться на том, что имеет отношение к делу, повышая общую точность.

Примеры:

ДО:

Напишите в блоге 3 абзаца о 5 лучших игровых приставках. Заметка должна быть информативной и увлекательной, а также написана в разговорном стиле.

НЕЛЬЗЯ:

Создайте статью в блоге о приставках для видеоигр.

## Используйте инструкции вместо ограничений

Инструкции и ограничения используются в подсказках, чтобы направлять вывод LLM.

- **Инструкция** содержит четкие указания относительно желаемого формата, стиля или содержания ответа. Она направляет модель на то, что она должна сделать или произвести.
- **Ограничение** - это набор ограничений или границ для ответа. Они ограничивают то, что модель не должна делать или избегать.



Растущие исследования показывают, что концентрация внимания на позитивных инструкциях в подсказках может быть более эффективной, чем опора на ограничения. Такой подход согласуется с тем, что люди предпочитают позитивные инструкции, а не списки того, что не следует делать.

Инструкции напрямую сообщают желаемый результат, в то время как ограничения могут оставить модель в догадках о том, что разрешено. Инструкции придают гибкость и поощряют творчество в определенных границах, в то время как ограничения могут ограничить потенциал модели. Кроме того, список ограничений может противоречить друг другу.

Ограничения все еще ценны, но в определенных ситуациях. Чтобы предотвратить генерацию моделью вредного или предвзятого контента или когда требуется строгий формат или стиль вывода.

По возможности используйте позитивные инструкции: вместо того чтобы говорить модели, что не нужно делать, скажите ей, что нужно делать. Это поможет избежать путаницы и повысит точность результатов.

ДО:

Напишите в блоге 1 абзац о 5 лучших игровых приставках. Укажите только консоль, компанию-производителя, год выпуска и общий объем продаж.

НЕЛЬЗЯ:

Напишите в блоге 1 абзац о 5 лучших игровых приставках. Не перечисляйте названия видеоигр.

В качестве наилучшей практики стафируйте приоритеты инструкций, четко указывая, что вы хотите, чтобы делала модель, и используйте ограничения только в тех случаях, когда это необходимо для обеспечения безопасности, ясности или выполнения конкретных требований. Экспериментируйте и проверяйте различные комбинации инструкций и ограничений, чтобы найти то, что лучше всего работает для ваших конкретных задач, и документируйте их.

## Контроль максимальной длины токена

Чтобы контролировать длину генерируемого LLM-ответа, вы можете либо установить ограничение на максимальное количество токенов в конфигурации, либо явно запросить определенную длину в запросе. Например:

*"Объясните квантовую физику в сообщении длиной в твит".*

## Использование переменных в подсказках

Чтобы повторно использовать подсказки и сделать их более динамичными, используйте переменные в подсказке, которые можно менять для разных вводимых данных. Например, как показано в табл. 20, подсказка выдает факты о городе. Вместо того чтобы жестко кодировать название города в подсказке, используйте переменную. Переменные помогут вам сэкономить время и силы, поскольку позволят избежать повторений. Если вам нужно использовать одну и ту же информацию в нескольких подсказках, вы можете сохранить ее в переменной и затем ссылаться на нее в каждой подсказке. Это очень удобно при интеграции подсказок в собственные приложения.

Справка	<p>ПЕРЕМЕННЫЕ</p> <p>{город} = "Амстердам"</p> <p>ПРОМПТ</p> <p>Вы – . Расскажите мне один факт о городе: {город}</p>
Выход	<p>Амстердам - прекрасный город, полный каналов, мостов и узких улочек. Это прекрасное место для посещения благодаря богатой истории, культуре и ночной жизни.</p>

Таблица 20. Использование переменных в подсказках

## Экспериментируйте с форматами ввода и стилями письма

Разные модели, конфигурации моделей, форматы подсказок, выбор слов и сабмитов могут давать разные результаты. Поэтому важно экспериментировать с такими атрибутами подсказки, как стиль, выбор слова и тип подсказки (нулевой выстрел, несколько выстрелов, системная подсказка).

Например, задание с целью создания текста о революционной игровой приставке Sega Dreamcast может быть сформулировано как **вопрос**, **утверждение** или **инструкция**, что приведет к различным результатам:

- **Вопросы:** Что представляла собой Sega Dreamcast и почему она стала такой революционной консолью?
- **Заявление:** Sega Dreamcast - игровая приставка шестого поколения, выпущенная компанией Sega в 1999 году. Она...
- **Инструкция:** Напишите один абзац, в котором опишите консоль Sega Dreamcast и объясните, почему она была настолько революционной.

## Для получения нескольких подсказок с заданиями на классификацию смешивайте классы

Вообще говоря, порядок следования примеров с несколькими снимками не должен иметь большого значения. Однако при выполнении задач классификации убедитесь, что вы смешиваете возможные классы ответов в примерах с несколькими снимками. Это связано с тем, что в противном случае вы можете слишком сильно привязаться к определенному порядку примеров. Смешивая возможные классы ответов, вы можете убедиться, что модель учится определять ключевые особенности каждого класса, а не просто запоминает порядок примеров. Это приведет к более надежной и обобщенной работе на невидимых данных.

Хорошее эмпирическое правило заключается в том, чтобы начать с 6 примеров нескольких выстрелов и затем проверить точность.

## Адаптация к обновлениям модели

Вам важно быть в курсе изменений в архитектуре модели, добавленных данных и возможностей. Пробуйте новые версии модели и корректируйте свои подсказки, чтобы лучше использовать новые возможности модели. Такие инструменты, как Vefiex AI Studio, отлично подходят для хранения, тестирования и документирования различных версий ваших подсказок.

## Экспериментируйте с форматами вывода

Помимо формата ввода, поэкспериментируйте с форматом вывода. Для некреативных задач, таких как извлечение, выборка, разбор, упорядочивание, ранжирование или категоризация данных, попробуйте вернуть выходные данные в структурированном формате, например JSON или XML.

Возвращение JSON-объектов из запроса, извлекающего данные, имеет некоторые преимущества. В реальном приложении мне не нужно вручную создавать этот формат JSON, я уже могу возвращать данные в порядке убывания (очень удобно при работе с объектами времени даты), но самое главное - запрос на формат JSON заставляет модель создавать структуру и ограничивает галлюцинации.

Итак, преимущества использования JSON для вывода данных:

- Возвращение всегда в одном стиле
- Сосредоточьтесь на данных, которые вы хотите получить

- Меньше вероятность галлюцинаций
- Сделайте его ориентированным на отношения
- Вы получаете типы данных
- Вы можете софировать его

В таблице 4 в разделе "Несколько подсказок" приведен пример возврата структурированного вывода.

## Восстановление JSON

Хотя возврат данных в формате JSON имеет множество преимуществ, он не лишен недостатков. Структурированная природа JSON, хотя и удобна для разбора и использования в приложениях, требует значительно большего количества лексем, чем обычный текст, что приводит к увеличению времени обработки и повышению стоимости. Кроме того, многословность JSON может легко занять все окно вывода, что становится особенно проблематичным, когда генерация резко обрывается из-за ограничений на количество токенов. Такое усечение часто приводит к недействительному JSON, в котором отсутствуют важные закрывающие скобки или брекеты, что делает вывод непригодным для использования. К счастью, такие инструменты, как библиотека `json-repair` (доступна на PyPI), могут оказаться бесценными в подобных ситуациях. Эта библиотека интеллектуально пытается автоматически исправить неполные или неправильно сформированные объекты JSON, что делает ее важным союзником при работе с JSON, сгенерированным LLM, особенно при решении потенциальных проблем с усечением.

## Работа со схемами

Использование структурированного JSON в качестве выходных данных - отличное решение, как мы уже неоднократно видели в этой статье. Но как быть с *входными данными*? Хотя JSON отлично подходит для структурирования *выходных данных*, генерируемых LLM, он также может быть невероятно полезен для структурирования *входных данных*, которые вы предоставляете. Здесь помощь приходят схемы JSON. Схема JSON определяет ожидаемую структуру и типы данных входных данных JSON. Предоставляя схему, вы даете LLM четкий план данных, которые он должен ожидать, помогая ему сфокусировать свое *внимание* на нужной информации и снижая риск неправильной интерпретации входных данных. Кроме того, схемы могут помочь установить взаимосвязи между различными частями данных и даже сделать LLM "ориентированным на время", включив поля даты или временной метки с определенными форматами.

Вот простой пример:

Допустим, вы хотите использовать LLM для создания описаний товаров в каталоге электронной коммерции. Вместо того чтобы просто предоставлять текстовое описание продукта в свободной форме, вы можете использовать схему JSON для определения атрибутов продукта:

```
{
  "тип": "object",
  "properties": {
    "name": { "type": "string", "description": "Название продукта" }, "category": {
    "type": "string", "description": "Категория товара" }, "цена": { "type":
    "число", "формат": "float", "description": "Продукт
цена" },
    "features": {
      "type": "array",
      "items": { "type": "string" }, "description":
      "Ключевые особенности продукта".
    },
    "release_date": { "type": "string", "format": "дата", "описание": "Дата
выпуска продукта" }.
  },
}
```

Фрагмент 5. Определение схемы структурированного вывода

Затем вы можете предоставить фактические данные о продукте в виде объекта JSON, который этой схеме:

```
{
  "name": "Беспроводные наушники",
  "категория": "Электроника", "цена":
  99.99,
  "features": ["Шумоподавление", "Bluetooth 5.0", "20-часовое время автономной
  работы"],
  "release_date": "2023-10-27"
```

Фрагмент 6. Структурированный вывод LLM

Предварительно обработав данные и предоставив вместо полных документов только схему и данные, вы даете LLM четкое представление об атрибутах продукта, включая дату его выпуска, что значительно повышает вероятность создания точного и релевантного описания. Такой структурированный подход к вводу данных, направляющий внимание LLM на соответствующие поля, особенно ценен при работе с большими объемами данных или при интеграции LLM в сложные приложения.

## Экспериментируйте вместе с другими оперативными инженерами

Если вы попали в ситуацию, когда нужно попытаться придумать хорошую подсказку, возможно, вам стоит найти несколько человек, которые попытаются это сделать. Если все будут следовать лучшим практикам (перечисленным в этой главе), вы увидите разницу в производительности между разными попытками.

## Передовой опыт CoT

Для подсказок CoT размещение ответа после рассуждений необходимо, поскольку генерация рассуждений изменяет лексемы, которые получает модель, предсказывая окончательный ответ.

С помощью CoT и самосогласованности вы должны уметь извлекать окончательный ответ из подсказки, отделяя его от рассуждений.

Для запроса CoT установите температуру на 0.

Цепочка подсказок основана на жадном декодировании, предсказывающем следующее слово в последовательности на основе наибольшей вероятности, присвоенной языковой моделью. Как правило, при использовании рассуждений, чтобы прийти к окончательному ответу, скорее всего, существует один единственный правильный ответ. Поэтому температура всегда должна быть равна 0.

## Документирование различных оперативных афлемптов

Последний совет уже упоминался в этой главе, но мы не можем не подчеркнуть, насколько он важен: документируйте свои попытки подсказок во всех подробностях, чтобы со временем понять, что получилось, а что нет.

Выводы подсказок могут отличаться в разных моделях, в разных настройках выборки и даже в разных версиях одной и той же модели. Более того, даже в идентичных подсказках для одной и той же модели могут возникать небольшие различия в оформлении предложений и выборе слов. (Например, как уже упоминалось, если два лексем имеют одинаковую предсказанную вероятность, то ничья может быть нарушена случайным образом. Это может повлиять на последующие предсказанные лексемы).



Мы рекомендуем создать лист Google с таблицей 21 в качестве шаблона. Преимущества такого подхода в том, что у вас будет полная запись, когда вам неизбежно придется вернуться к своей работе с подсказками - либо для того, чтобы повторить ее в будущем (вы удивитесь, как много вы можете забывать после перерыва на шофи), для проверки быстродействия на разных версиях модели, а также для отладки будущих ошибок.

Помимо полей в этой таблице, полезно также отслеживать версию подсказки (итерацию), поле для записи результата "ОК/НЕТ ОК/КОМЕТИМО ОК" и поле для записи обратной связи. Если вам посчастливилось использовать Vefiex AI Studio, сохраните свои подсказки (с помощью функции имя и версию, указанные в документации) и отследить гиперссылку на сохраненную подсказку в таблице. Таким образом, вы всегда будете в одном клике от повторного запуска подсказок.

Работая над системой *генерации расширенного поиска*, вы также должны фиксировать конкретные аспекты системы RAG, которые влияют на то, какой контент был введен в подсказку, включая запрос, настройки чанков, вывод чанков и другую информацию.

Когда вы почувствуете, что подсказка близка к идеалу, перенесите ее в кодовую базу вашего проекта. А в кодовой базе сохраняйте подсказки в отдельном от кода файле, чтобы их было проще поддерживать. Наконец, в идеале ваши подсказки - это паффи операционализированной системы, и как инженер подсказок вы должны полагаться на автоматизированные тесты и процедуры оценки, чтобы понять, насколько хорошо ваша подсказка подходит к задаче.

Разработка подсказок - это итеративный процесс. Создавайте и тестируйте различные подсказки, анализируйте и документируйте результаты. Доработайте подсказку, основываясь на результатах работы модели. Продолжайте экспериментировать, пока не добьетесь желаемого результата. Когда вы меняете модель или конфигурацию модели, возвращайтесь и продолжайте экспериментировать с ранее использованными подсказками.

Имя	[название и версия вашей программы]		
Цель	[Объяснение цели этой попытки в одном предложении]		
Модель	[название и версия используемой модели]		
Температура	[значение между 0 и 1]	Лимит жетонов	[номер]
Топ-K	[номер]	Топ-P	[номер]
Справка	[Напишите полный текст подсказки] .		
Выход	[Записать выход или несколько выходов]		

Таблица 21. Шаблон для документирования подсказок

## Резюме

В этом обзоре рассматривается разработка подсказок. Мы изучили различные техники создания подсказок, такие как:

- Нулевая подсказка
- Несколько подсказок по выстрелам
- Системные подсказки
- Ролевые подсказки
- Контекстные подсказки
- Пошаговая подсказка
- Цепочка мыслей
- Самостоятельная последовательность
- Дерево мыслей

- ReAct

Мы даже рассмотрели способы автоматизации подсказок.

Далее в статье обсуждаются проблемы искусственного интеллекта, например, проблемы, которые могут возникнуть, если ваши подсказки окажутся недостаточными. В заключение мы привели лучшие практики, как стать лучшим инженером по подсказкам.

## Примечания

1. Google, 2023, Gemini by Google. Доступно по адресу: <https://gemini.google.com>.
2. Google, 2024, Gemini for Google Workspace Prompt Guide. Available at: <https://inthecloud.withgoogle.com/gemini-for-google-workspace-prompt-guide/dl-cd.html>.
3. Google Cloud, 2023, Введение в подсказки. Доступно по адресу: <https://cloud.google.com/vertex-ai/generative-ai/docs/learn/prompts/introduction-prompt-design>.
4. Google Cloud, 2023, тело запроса текстовой модели: методы выборки Top-P и Top-K. Доступно по адресу: [https://cloud.google.com/vertex-ai/docs/generative-ai/model-reference/text#request\\_body](https://cloud.google.com/vertex-ai/docs/generative-ai/model-reference/text#request_body).
5. Wei, J., et al., 2023, Zero Shot - Fine Tuned language models are zero shot learners. Доступно по адресу: <https://arxiv.org/pdf/2109.01652.pdf>.
6. Google Cloud, 2023, Google Cloud Model Garden. Доступно по адресу: <https://cloud.google.com/model-garden>.
7. Браун, Т., и др., 2023, Few Shot - языковые модели для малоопытных учеников. Available at: <https://arxiv.org/pdf/2005.14165.pdf>.
8. Чжэн, Л., и др., 2023, Сделай шаг назад: Evoking Reasoning via Abstraction in Large Language Models. Доступно по адресу: <https://openreview.net/pdf?id=3bq3jsvcQ1>
9. Wei, J., et al., 2023, Chain of Thought Prompting. Available at: <https://arxiv.org/pdf/2201.11903.pdf>.
10. Google Cloud Platform, 2023, Chain of Thought и React. Доступно по адресу: [https://github.com/GoogleCloudPlatform/generative-ai/blob/main/language/prompts/examples/chain\\_of\\_thought\\_react.ipynb](https://github.com/GoogleCloudPlatform/generative-ai/blob/main/language/prompts/examples/chain_of_thought_react.ipynb).
11. Wang, X., et al., 2023, Self Consistency Improves Chain of Thought reasoning in language models. Доступно по адресу: <https://arxiv.org/pdf/2203.11171.pdf>.
12. Яо, С., и др., 2023, Дерево мыслей: Deliberate Problem Solving with Large Language Models. Доступно по адресу: <https://arxiv.org/pdf/2305.10601.pdf>.
13. Яо, С., и др., 2023, ReAct: Synergizing Reasoning and Acting in Language Models. Available at: <https://arxiv.org/pdf/2210.03629.pdf>.
14. Google Cloud Platform, 2023, Предварительное наведение справок: Chain of Thought and React. Доступно по адресу: [https://github.com/GoogleCloudPlatform/applied-ai-engineering-samples/blob/main/genai-on-vertex-ai/advanced\\_prompting\\_training/cot\\_react.ipynb](https://github.com/GoogleCloudPlatform/applied-ai-engineering-samples/blob/main/genai-on-vertex-ai/advanced_prompting_training/cot_react.ipynb).
15. Zhou, C., et al., 2023, Automatic Prompt Engineering - Large Language Models are Human-Level Prompt Engineers. Доступно по адресу: <https://arxiv.org/pdf/2211.01910.pdf>.