Часть 1: основы и базовые техники

От переводчика

Представляю вашему вниманию перевод статьи "Prompt Engineering" (Промпт-инжиниринг) авторства Lee Boonstra - Software Engineer Tech Lead, Office of the CTO в Google. Это первая часть из планируемого цикла из трех статей, поскольку оригинальный документ весьма объёмен (68 страниц) и насыщен полезной информацией.

Важно отметить, что оригинальная публикация фокусируется в основном на моделях Gemini и сервисе Vertex AI от Google, однако описанные принципы, техники и советы универсальны и применимы практически ко всем современным большим языковым моделям (GPT, Claude, Llama и др.). Концепции промпт-инжиниринга остаются актуальными независимо от конкретной используемой модели.

Обратите внимание: Это неофициальный перевод, выполненный энтузиастом с целью сделать ценную информацию доступной русскоязычному сообществу. Результаты работы моделей с переведенными примерами промптов могут отличаться от тех, что приведены в оригинальной статье. Это связано с различиями в обработке текстов на разных языках большими языковыми моделями.

Некоторые технические термины оставлены без перевода или транслитерированы (например, "промпт", "промпт-инжиниринг", "Top-K", "Top-P"), поскольку они уже вошли в профессиональный жаргон русскоязычных специалистов в сфере ИИ.

Запланированные части цикла:

- **Часть 1 (текущая)**: Основы промпт-инжиниринга и базовые техники
- **Часть 2**: Продвинутые техники промптинга и работа с кодом
- **Часть 3**: Лучшие практики и рекомендации

Публикация последующих частей будет зависеть от интереса сообщества к данной теме. В комментариях буду рад обсудить как вопросы перевода, так и сами техники промптинга, поделиться опытом и узнать о ваших находках в этой области.

Надеюсь, этот перевод поможет вам лучше понять принципы работы с большими языковыми моделями и создавать более эффективные промпты для своих задач.

Приятного чтения!

> Для создания промптов не требуется быть специалистом по данным или разработчиком систем машинного обучения – с этим справится каждый.

Введение

Рассматривая работу большой языковой модели, мы видим, что текстовый промпт (иногда дополненный другими типами данных, например изображениями) служит входной информацией, на основе которой модель формирует определенный результат. Важно понимать: для создания промптов не требуется быть специалистом по данным или разработчиком систем машинного обучения – с этим справится каждый. Однако разработка по-настоящему эффективного промпта – дело непростое. На его результативность влияет множество факторов: выбранная модель, данные её обучения, настройки, подбор слов, стилистика, структура и контекст. Именно поэтому промпт-инжиниринг представляет собой процесс постоянного совершенствования. Неудачно составленные промпты приводят к неоднозначным или неточным ответам, снижая способность модели генерировать полезные результаты.

Общаясь с чат-ботом Gemini[1], вы фактически уже пишете промпты. Однако в этой статье основное внимание уделяется созданию промптов для модели Gemini в экосистеме Vertex AI или через API, поскольку при прямом взаимодействии с моделью вы получаете контроль над такими параметрами как температура и другие настройки.

В этой статье мы подробно рассмотрим промпт-инжиниринг. Мы познакомим вас с различными техниками составления промптов, поможем сделать первые шаги и поделимся советами и лучшими практиками для достижения мастерства. Также мы обсудим трудности, с которыми вы можете столкнуться при создании промптов.

Промпт-инжиниринг

Важно понимать принцип работы БЯМ: по сути, это механизм предсказания. Модель получает последовательность текста и предсказывает, каким должен быть следующий токен, основываясь на данных своего обучения. БЯМ повторяет этот процесс многократно, добавляя предсказанный токен к имеющейся последовательности и продолжая прогнозировать дальше. Предсказание строится на связи между содержимым предыдущих токенов и информацией, полученной моделью во время обучения.

Создавая промпт, вы стремитесь настроить БЯМ так, чтобы она выдала правильную последовательность токенов. Промпт-инжиниринг – это процесс разработки качественных промптов, направляющих БЯМ к созданию точных результатов. Он включает в себя эксперименты по поиску оптимальных формулировок, настройку длины промпта и оценку его стиля и структуры применительно к конкретной задаче. В контексте обработки естественного языка и работы с БЯМ, промпт – это входные данные, подаваемые модели для получения ответа или прогноза.

Такие промпты можно использовать для решения различных задач понимания и генерации, включая суммаризацию текста, извлечение информации, вопросно-ответные системы, классификацию текста, перевод языков или кода, генерацию кода, документирование кода или логические рассуждения.

Рекомендуем ознакомиться с руководствами Google[2,3] по составлению промптов, содержащими простые и эффективные примеры.

При работе с промпт-инжинирингом первым шагом является выбор модели. Промпты могут требовать оптимизации под конкретную модель, независимо от того, используете ли вы языковые модели Gemini в Vertex AI, GPT, Claude или открытые модели типа Gemma или LLaMA.

Помимо самого промпта, вам также потребуется настраивать различные конфигурации БЯМ.

Настройка вывода БЯМ

После выбора модели вам нужно определиться с её конфигурацией. Большинство БЯМ предлагают различные параметры настройки, контролирующие вывод. Эффективный промпт-инжиниринг требует оптимальной настройки этих параметров для вашей задачи.

Длина вывода

Важным параметром конфигурации является количество токенов, генерируемых в ответе. Генерация большего количества токенов требует больших вычислительных ресурсов от БЯМ, что приводит к повышенному энергопотреблению, потенциально более длительному времени отклика и более высоким затратам.

Уменьшение длины вывода БЯМ не делает вывод модели стилистически или текстуально более лаконичным, оно просто заставляет модель прекратить предсказание после достижения установленного лимита. Если вам нужна короткая длина вывода, возможно, придется также адаптировать промпт соответствующим образом.

Ограничение длины вывода особенно важно для некоторых техник промптинга, таких как ReAct, где БЯМ может продолжать выдавать бесполезные токены после нужного вам ответа.

Помните, что генерация большего количества токенов требует больших вычислительных ресурсов от БЯМ, что ведет к повышенному энергопотреблению и потенциально более долгому времени ответа, а это, в свою очередь, повышает стоимость.

Параметры сэмплирования

БЯМ не предсказывают строго один токен. Вместо этого они прогнозируют вероятности для возможных следующих токенов, где каждый токен в словаре БЯМ получает определённую вероятность. Затем из этих вероятностей токенов выбирается один, который становится следующим выходным токеном. Temperature, Top-K и Top-P – наиболее распространенные параметры конфигурации, определяющие, как прогнозируемые вероятности токенов обрабатываются для выбора единственного выходного токена.

Температура

Температура (Temperature) контролирует степень случайности при выборе токена. Более низкие температуры хороши для промптов, где ожидается более детерминированный ответ, в то время как более высокие температуры могут привести к более разнообразным или неожиданным результатам. Температура 0 (жадное декодирование) является детерминированной: всегда выбирается токен с наивысшей вероятностью (хотя стоит отметить, что если два токена имеют одинаковую наивысшую прогнозируемую вероятность, в зависимости от реализации разрешения таких ситуаций вы можете не всегда получать одинаковый результат даже при температуре 0).

Настройка подбирается ближе к максимальной, приводит к более случайному выводу. И по мере роста температуры все токены становятся одинаково вероятными кандидатами на роль следующего предсказанного токена.

Управление температурой в Gemini можно понимать схожим образом с функцией softmax, используемой в машинном обучении. Низкая настройка температуры соответствует низкой температуре softmax (T), выделяя одну предпочтительную температуру с высокой определенностью. Более высокая настройка температуры Gemini подобна высокой температуре softmax, делая приемлемым более широкий диапазон температур вокруг выбранного значения. Эта повышенная неопределенность подходит для сценариев, где жесткая, точная температура может не быть существенной, например, при экспериментах с творческими результатами.

Top-K и Top-P

Top-K и Top-P (также известный как nucleus sampling)[4] – это два параметра сэмплирования, используемые в БЯМ для ограничения предсказанного следующего токена токенами с наивысшими прогнозируемыми вероятностями. Как и температура, эти параметры сэмплирования контролируют случайность и разнообразие генерируемого текста.

- **Top-K** сэмплирование выбирает K наиболее вероятных токенов из прогнозируемого распределения модели. Чем выше Top-K, тем более творческим и разнообразным будет вывод модели; чем ниже Top-K, тем более сдержанным и фактическим будет вывод модели. Top-K, равный 1, эквивалентен жадному декодированию.
- **Top-P** сэмплирование выбирает токены с наивысшей вероятностью, суммарная вероятность которых не превышает определенного значения (P). Значения P варьируются от 0 (жадное декодирование) до 1 (все токены в словаре БЯМ).

Лучший способ выбрать между Top-K и Top-P – экспериментировать с обоими методами (или их комбинацией) и определить, какой из них дает результаты, наиболее соответствующие вашим целям.

Всё вместе

Выбор между Top-K, Top-P, температурой и количеством генерируемых токенов зависит от конкретного приложения и желаемого результата, при этом все эти настройки влияют друг на друга. Также важно понимать, как выбранная модель комбинирует различные параметры сэмплирования.

Если в Vertex Studio доступны температура, Top-K и Top-P, токены, удовлетворяющие критериям как Top-K, так и Top-P, становятся кандидатами на следующий предсказанный токен, а затем применяется температура для выборки из токенов, прошедших критерии Top-K и Top-P. Если доступен только Top-K или Top-P, логика такая же, но используется только один параметр Top-K или P.

Если температура недоступна, любые токены, соответствующие критериям Top-K и/или Top-P, затем случайно выбираются для получения единственного следующего предсказанного токена.

При экстремальных значениях одного параметра сэмплирования этот параметр либо отменяет другие настройки конфигурации, либо становится несущественным.

- Если вы установите температуру на 0, Top-K и Top-P становятся несущественными – токен с наивысшей вероятностью становится следующим предсказанным. Если вы установите температуру на очень высокое значение (выше 1, обычно десятки), температура становится несущественной, и любые токены, прошедшие критерии Top-K и/или Top-P, случайно выбираются для определения следующего предсказанного токена.
- Если вы установите Top-K равным 1, температура и Top-P становятся несущественными. Только один токен проходит критерии Top-K, и именно этот токен становится следующим предсказанным. Если установить Top-K на очень высокое значение, например, на размер словаря БЯМ, любой токен с ненулевой вероятностью быть следующим будет соответствовать критериям Top-K, и ни один из них не будет отсеян.
- Если вы установите Top-P на 0 (или очень малое значение), большинство реализаций сэмплирования БЯМ будут рассматривать только токен с наивысшей

вероятностью как удовлетворяющий критериям Top-P, делая температуру и Top-K несущественными. Если установить Top-P на 1, любой токен с ненулевой вероятностью быть следующим будет соответствовать критериям Top-P, и ни один из них не будет отсеян.

В качестве общей отправной точки, температура 0.2, Top-P 0.95 и Top-K 30 дадут относительно связные результаты, которые могут быть творческими, но не чрезмерно. Если вам нужны особенно творческие результаты, попробуйте начать с температуры 0.9, Top-P 0.99 и Top-K 40. А если вы хотите менее творческие результаты, попробуйте начать с температуры 0.1, Top-P 0.9 и Top-K 20. Наконец, если ваша задача всегда имеет единственно правильный ответ (например, решение математической задачи), начните с температуры 0.

ПРИМЕЧАНИЕ: С большей свободой (более высокие значения температуры, Top-K, Top-P и количества выходных токенов) БЯМ может генерировать текст, который становится менее релевантным.

ПРЕДУПРЕЖДЕНИЕ: Встречались ли вам ответы, заканчивающиеся большим количеством слов-заполнителей? Это известно как "баг зацикливания повторений", распространенная проблема в больших языковых моделях, когда модель застревает в цикле, многократно генерируя одно и то же (заполняющее) слово, фразу или структуру предложения. Это часто усугубляется неподходящими настройками температуры и Top-K/Top-P. Такие зацикливания могут возникать как при низких, так и при высоких настройках температуры, хотя по разным причинам. При низких температурах модель становится чрезмерно детерминированной, жестко придерживаясь пути с наивысшей вероятностью, что может привести к циклу, если этот путь возвращается к ранее сгенерированному тексту. Напротив, при высоких температурах вывод модели становится излишне случайным, увеличивая вероятность того, что случайно выбранное слово или фраза по совпадению приведет обратно к предыдущему состоянию, создавая цикл из-за огромного числа доступных вариантов. В обоих случаях процесс сэмплирования модели "застревает", давая монотонный и бесполезный вывод, пока не заполнится выходное окно. Решение часто требует тщательной настройки значений температуры и Top-K/Top-P для нахождения оптимального баланса между детерминизмом и случайностью.

Техники промптинга

Большие языковые модели настроены на следование инструкциям и обучены на огромных объемах данных, благодаря чему они способны понимать промпт и генерировать ответ. Однако БЯМ не идеальны; чем четче сформулирован текст промпта, тем легче модели предсказать наиболее вероятный следующий текст. Кроме того, определенные техники, использующие особенности обучения и функционирования БЯМ, помогут вам получить более релевантные результаты.

Теперь, когда мы понимаем, что такое промпт-инжиниринг и что для него требуется, давайте рассмотрим примеры наиболее важных техник промптинга.

Общий промптинг / промптинг с нулевым примером

Промпт с нулевым примером (zero-shot)[5] – это простейший тип промпта. Он содержит только описание задачи и некоторый текст, с которого БЯМ может начать. Этот ввод может быть чем угодно: вопросом, началом истории или инструкциями. Название "zero-shot" означает 'без примеров'.

Воспользуемся Vertex AI Studio (для языка) в Vertex AI[6], который предоставляет площадку для тестирования промптов. В Таблице 1 вы увидите пример промпта с нулевым примером для классификации отзывов о фильмах.

Формат таблицы, использованный ниже, отлично подходит для документирования промптов. Ваши промпты, вероятно, пройдут множество итераций, прежде чем они попадут в кодовую базу, поэтому важно отслеживать работу по промпт-инжинирингу дисциплинированным, структурированным образом. Подробнее об этом формате таблицы, важности отслеживания работы по промпт-инжинирингу и процессе разработки промптов рассказано в разделе "Лучшие практики" ниже ("Документирование различных попыток промптинга").

Температуру модели следует установить на низкое значение, поскольку креативность здесь не требуется, и мы используем стандартные значения gemini-pro для Top-K и Top-P, что фактически отключает обе настройки (см. "Настройка вывода БЯМ" выше). Обратите внимание на сгенерированный вывод. Слова "*disturbing*" *(тревожный)* и "*masterpiece*" *(шедевр)* должны сделать предсказание немного сложнее, так как оба слова используются в одном предложении.

Название	1_1_movie_classification
Цель	Классифицировать отзывы о фильмах как положительные, нейтральные или отрицательные.
Модель	gemini-pro
Температура	0.1
Top-K	N/A
Промпт	Классифицируй отзывы о фильмах как ПОЛОЖИТЕЛЬНЫЕ, НЕЙТРАЛЬНЫЕ или ОТРИЦАТЕЛЬНЫЕ. Отзыв: "Она" - тревожное исследование, раскрывающее направление, в котором движется человечество, если позволить ИИ продолжать развиваться бесконтрольно. Хотел бы я, чтобы было больше таких шедевров. Отношение:
Вывод	ПОЛОЖИТЕЛЬНЫЙ

Таблица 1. Пример промптинга с нулевым примером

Когда промптинг с нулевым примером не работает, можно предоставить демонстрации или примеры в промпте, что приводит к промптингу "с одним примером" и "с несколькими примерами".

Промптинг с одним и несколькими примерами

При создании промптов для моделей ИИ полезно предоставлять примеры. Эти примеры помогают модели понять, что именно вы запрашиваете. Примеры особенно полезны, когда вы хотите направить модель к определенной структуре или шаблону вывода.

Промпт с одним примером (one-shot) предоставляет *единственный* пример, отсюда и название. Идея в том, что у модели есть образец, которому она может подражать для наилучшего выполнения задачи.

Промпт с несколькими примерами (few-shot)[7] предоставляет модели *несколько* примеров. Подход схож с промптингом с одним примером, но множественные примеры желаемого шаблона повышают вероятность того, что модель будет следовать этому шаблону.

Количество примеров, необходимых для промптинга с несколькими примерами, зависит от ряда факторов, включая сложность задачи, качество примеров и возможности используемой модели генеративного ИИ. В качестве общего правила, для промптинга с несколькими примерами следует использовать не менее трех-пяти примеров. Однако для более сложных задач может потребоваться больше примеров, или, наоборот, меньше из-за ограничений на длину входных данных вашей модели.

Таблица 2 демонстрирует пример промпта с несколькими примерами, давайте использовать те же настройки конфигурации модели gemini-pro, что и раньше, лишь увеличив лимит токенов для размещения более длинного ответа.

Цель	Преобразовать заказы пиццы в JSON
Модель	gemini-pro
Температура	0.1
Top-K	N/A

Промпт	Преобразуй заказ пиццы клиента в корректный JSON: ПРИМЕР: Я хочу маленькую пиццу с сыром, томатным соусом и пепперони. JSON-ответ:{ "size": "small", "type": "normal", "ingredients": [["cheese", "tomato sauce", "peperoni"]] } ПРИМЕР: Можно мне большую пиццу с томатным соусом, базиликом и моцареллой { "size": "large", "type": "normal", "ingredients": [["tomato sauce", "bazel", "mozzarella"]] } Теперь я хотел бы большую пиццу, одна половина с сыром и моцареллой. А другая с томатным соусом, ветчиной и ананасом. JSON-ответ:
Вывод	{ "size": "large", "type": "half-half", "ingredients": [["cheese", "mozzarella"], ["tomato sauce", "ham", "pineapple"]] }

Таблица 2. Пример промптинга с несколькими примерами

При выборе примеров для промпта используйте примеры, релевантные задаче, которую вы хотите выполнить. Примеры должны быть разнообразными, высокого качества и хорошо составленными. Даже небольшая ошибка может запутать модель и привести к нежелательному результату.

Если вы пытаетесь генерировать результаты, устойчивые к разнообразным входным данным, важно включать граничные случаи в ваши примеры. Граничные случаи – это входные данные, которые необычны или неожиданны, но с которыми модель всё равно должна уметь справляться.

Системный, контекстуальный и ролевой промптинг

Системный, контекстуальный и ролевой промптинг – это техники, используемые для направления генерации текста БЯМ, но они фокусируются на разных аспектах:

- **Системный промптинг** задает общий контекст и цель для языковой модели. Он определяет "общую картину" того, что модель должна делать, например, переводить текст, классифицировать отзыв и т.д.
- **Контекстуальный промптинг** предоставляет конкретные детали или фоновую информацию, относящуюся к текущему разговору или задаче. Он помогает модели понять нюансы запроса и соответственно адаптировать ответ.
- **Ролевой промптинг** назначает языковой модели конкретную роль или идентичность. Это помогает модели генерировать ответы, соответствующие заданной роли и связанным с ней знаниям и поведению.

Между системным, контекстуальным и ролевым промптингом может быть значительное пересечение. Например, промпт, который назначает роль системе, также может иметь контекст.

Однако каждый тип промпта служит немного разным основным целям:

- Системный промпт: Определяет фундаментальные возможности и всеобъемлющую цель модели.
- Контекстуальный промпт: Предоставляет немедленную, специфичную для задачи информацию для направления ответа. Он узкоспецифичен для текущей задачи или входных данных, которые динамически меняются.
- Ролевой промпт: Формирует стиль и голос вывода модели. Он добавляет слой конкретики и индивидуальности.

Различение между системными, контекстуальными и ролевыми промптами предоставляет структуру для проектирования промптов с четким намерением, позволяя гибкие комбинации и облегчая анализ того, как каждый тип промпта влияет на вывод языковой модели.

Давайте рассмотрим эти три различных типа промптов.

Системный промптинг

Таблица 3 содержит системный промпт, где я указываю дополнительную информацию о том, как вернуть результат. Я увеличил температуру для получения более высокого уровня креативности и установил более высокий лимит токенов. Однако благодаря четкой инструкции о том, как вернуть результат, модель не возвращает дополнительный текст.

Цель	Классифицировать отзывы о фильмах как положительные, нейтральные или отрицательные.
Модель	gemini-pro
Температура	0.1
Top-K	40
Промпт	Классифицируй отзывы о фильмах как положительные, нейтральные или отрицательные. Верни только метку прописными буквами. Отзыв: "Она" - тревожное исследование, раскрывающее направление, в котором движется человечество, если позволить ИИ продолжать развиваться бесконтрольно. Настолько тревожное, что я не смог досмотреть. Настроение:
Вывод	ОТРИЦАТЕЛЬНЫЙ

Таблица 3. Пример системного промптинга

Системные промпты могут быть полезны для генерации результатов, отвечающих конкретным требованиям. Название 'системный промпт' на самом деле означает 'предоставление дополнительной задачи системе'. Например, вы можете использовать системный промпт для генерации фрагмента кода, совместимого с определенным языком

программирования, или вы можете использовать системный промпт для возврата определенной структуры. Посмотрите на Таблицу 4, где я возвращаю результат в формате JSON.

Цель	Классифицировать отзывы о фильмах как положительные, нейтральные или отрицательные, вернуть JSON.
Модель	gemini-pro
Температура	0.1
Top-K	40
Промпт	Классифицируй отзывы о фильмах как положительные, нейтральные или отрицательные. Верни корректный JSON: Отзыв: "Она" - тревожное исследование, раскрывающее направление, в котором движется человечество, если позволить ИИ продолжать развиваться бесконтрольно. Настолько тревожное, что я не смог досмотреть. Схема: MOVIE: { "sentiment": String "POSITIVE" | "NEGATIVE" | "NEUTRAL", "name": String } MOVIE REVIEWS: { "movie_reviews": [MOVIE] } JSON Response:

Вывод	{
	"movie_reviews": [
	{
	"sentiment": "NEGATIVE", "name": "Her"
	}
	]
	}

Таблица 4. Пример системного промптинга с форматом JSON

Возврат данных в формате JSON объектов из промпта, извлекающего данные, имеет ряд преимуществ. В реальном приложении мне не нужно вручную создавать этот JSON-формат, я уже могу вернуть данные в отсортированном порядке (очень удобно при работе с объектами datetime), но что наиболее важно, запрашивая формат JSON, мы заставляем модель создавать структуру и ограничиваем галлюцинации.

Системные промпты также могут быть очень полезны для обеспечения безопасности и предотвращения токсичности. Для контроля вывода просто добавьте дополнительную строку к вашему промпту, например: 'Вы должны быть вежливы в своем ответе'.

Ролевой промптинг

Ролевой промптинг – это техника в промпт-инжиниринге, которая включает назначение определенной роли модели генеративного ИИ. Это может помочь модели генерировать

более релевантный и информативный вывод, поскольку модель может адаптировать свои ответы к конкретной роли, которая ей была назначена.

Например, вы можете назначить модели генеративного ИИ роль редактора книг, воспитателя детского сада или мотивационного спикера. После назначения модели роли вы можете давать ей промпты, специфичные для этой роли. Например, вы можете попросить учителя создать план урока, который он затем может просмотреть. Посмотрите на Таблицу 5, где модель принимает роль туристического гида.

Цель	Выступить в роли туристического гида и предоставить 3 предложения по путешествию
Модель	gemini-pro
Температура	1
Top-K	40
Промпт	Я хочу, чтобы ты выступил в роли туристического гида. Я буду писать тебе о своем местоположении, а ты предлагай 3 места для посещения рядом со мной. В некоторых случаях я также укажу тип мест, которые хочу посетить. Мое предложение: "Я в Амстердаме и хочу посетить только музеи." Предложения по путешествию:
Вывод	1. Рейксмузеум: Всемирно известный музей, славящийся голландскими шедеврами, такими как "Ночной дозор" Рембрандта. Не пропустите невероятную коллекцию картин и артефактов, представляющих голландское искусство и историю. 2. Музей Ван Гога: Погрузитесь в захватывающие работы Винсента ван Гога. Музей содержит крупнейшую коллекцию его картин и рисунков, включая "Звездную ночь" и "Подсолнухи". 3. Stedelijk Museum Amsterdam: Откройте для себя современное и актуальное искусство со всего мира. Коллекция музея включает работы Пикассо, Кандинского и Ритвельда, размещенные в эффектном современном здании.

Таблица 5. Пример ролевого промптинга

Приведенный выше пример демонстрирует принятие роли туристического агента. При изменении роли на преподавателя географии вы заметите, что получите другой ответ.

Определение ролевой перспективы для модели ИИ дает ей четкий план тона, стиля и сфокусированной экспертизы, которые вы ищете, чтобы улучшить качество, релевантность и эффективность результата.

Вот некоторые стили, которые вы можете выбрать и которые я считаю эффективными: Конфронтационный, Описательный, Прямолинейный, Формальный, Юмористический,

Влиятельный, Неформальный, Вдохновляющий, Убедительный

Давайте изменим наш промпт в Таблице 6, включив юмористический и вдохновляющий стиль.

Промпт	Я хочу, чтобы ты выступил в роли туристического гида. Я буду писать тебе о своем местоположении, а ты предлагай 3 места для посещения рядом со мной в юмористическом стиле. Мое предложение: "Я на Манхэттене." Предложения по путешествию:
Вывод	1. Узрите Имперское состояние ума: Поднимитесь на головокружительную высоту Эмпайр-стейт-билдинг и насладитесь видом на манхэттенский горизонт. Приготовьтесь почувствовать себя Кинг-Конгом на вершине Большого Яблока, минус банан размером с гигантскую обезьяну. 2. Станьте художественным чудаком в MoMA: Высвободите своего внутреннего ценителя искусства в Музее современного искусства (MoMA). Созерцайте шедевры, которые поразят ваш разум и заставят вас задуматься, имеют ли ваши рисунки-палочки какую-либо художественную ценность. 3. Шопинг до упаду на Пятой авеню: Побалуйте себя шопинг-терапией на легендарной Пятой авеню. Приготовьтесь к шоковым ценникам, разглядывая витрины (или совершая реальные покупки) в дизайнерских бутиках, от которых ваш кошелек заплачет. Но эй, вы же на Манхэттене, так что можете с таким же успехом окунуться в роскошь!

Таблица 6. Пример ролевого промптинга с юмористическим тоном и стилем

Контекстуальный промптинг

Предоставляя контекстуальные промпты, вы можете помочь обеспечить максимально плавное и эффективное взаимодействие с ИИ. Модель сможет быстрее понять ваш запрос и генерировать более точные и релевантные ответы, как вы можете видеть в примере Таблицы 7.

Цель	Предложить статьи для блога о ретро-играх
Модель	gemini-pro
Температура	1
Top-K	40

Промпт	Контекст: Ты пишешь для блога о ретро-аркадных видеоиграх 80-х. Предложи 3 темы для написания статьи с несколькими строками описания того, что должна содержать эта статья.
Вывод	1. **Эволюция дизайна аркадных автоматов:** - Эта статья исследовала бы эволюцию дизайна аркадных автоматов, от ранних деревянных и металлических кабинетов 1970-х годов до элегантных, неоново-освещенных дизайнов 1980-х и далее. 2. **Взрыв из прошлого: Культовые аркадные игры 80-х:** - Эта статья предложила бы взгляд на историю, влияние и наследие некоторых из самых знаковых и влиятельных аркадных игр 1980-х годов, подчеркивая их инновации, популярные механики и вечное очарование. 3. **Взлёт и ретро-возрождение пиксельной графики:** - Эта статья погрузилась бы в эволюцию пиксельной графики как определяющего визуального стиля ранних аркадных игр, прослеживая её корни, исследуя её техники и обсуждая возрождение пиксельной графики в современных играх и цифровом искусстве.

Таблица 7. Пример контекстуального промптинга

Источники и ссылки

Примечание переводчика: некоторые ссылки из оригинальной публикации не работают.

1. Google, 2023, Gemini by Google: https://gemini.google.com
2. Google, 2024, Gemini for Google Workspace Prompt Guide: https://inthecloud.withgoogle.com/gemini-for-google-workspace-prompt-guide/dl-cd.html
3. Google Cloud, 2023, Introduction to Prompting: https://cloud.google.com/vertex-ai/generative-ai/docs/learn/prompts/introduction-prompt-design
4. Google Cloud, 2023, Text Model Request Body: Top-P & top-K sampling methods: 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: https://arxiv.org/pdf/2109.01652.pdf
6. Google Cloud, 2023, Google Cloud Model Garden: https://cloud.google.com/model-garden

7. Brown, T., et al., 2023, Few Shot - Language Models are Few Shot learners: https://arxiv.org/pdf/2005.14165.pdf

---

Завершение первой части

Мы рассмотрели основы промпт-инжиниринга, включая базовые настройки больших языковых моделей (температура, Top-K, Top-P), а также познакомились с ключевыми техниками промптинга: промптинг с нулевым примером, с одним и несколькими примерами, системным, ролевым и контекстуальным промптингом.

Все эти методы составляют фундамент для эффективной работы с БЯМ, позволяя получать более точные, релевантные и полезные ответы на ваши запросы.

Что дальше?

В следующих частях цикла нас ждет погружение в более продвинутые техники и практические аспекты промпт-инжиниринга:

Во второй части мы рассмотрим:

- Промптинг с отступлением (Step-back prompting)
- Цепочку рассуждений (Chain of Thought)
- Техники самосогласованности (Self-consistency)
- Дерево рассуждений (Tree of Thoughts)
- Метод "Рассуждение и действие" (ReAct)
- Автоматический промпт-инжиниринг
- Работу с кодом через промпты: написание, объяснение, перевод и отладка кода

В третьей части нас ждут:

- Лучшие практики промпт-инжиниринга
- Методы предоставления примеров
- Проектирование простых и эффективных промптов
- Работа с JSON и схемами
- Документирование экспериментов с промптами
- ...и многое другое!

Надеюсь, эта первая часть была полезной и вызвала интерес к дальнейшему изучению темы. Буду рад вашим комментариям, вопросам и обсуждению опыта работы с техниками промптинга.

Revision #2
Created 2026-02-16 09:00:14 UTC by Эд Кукса
Updated 2026-02-16 09:07:59 UTC by Эд Кукса