Заключительная часть: лучшие практики и рекомендации

От переводчика

Представляю вашему вниманию заключительную, третью часть перевода статьи "Prompt Engineering" (Промпт-инжиниринг) авторства Lee Boonstra — Software Engineer Tech Lead, Office of the CTO в Google. Этот материал завершает цикл публикаций, посвященных эффективному взаимодействию с большими языковыми моделями.

В первой части мы познакомились с основами промпт-инжиниринга и базовыми техниками промптинга. Во второй части рассмотрели продвинутые методы, такие как промптинг с отступлением, цепочку рассуждений, самосогласованность, дерево рассуждений и ReAct. Третья часть, которую вы читаете сейчас, посвящена лучшим практикам и прикладным рекомендациям, которые помогут существенно повысить качество и эффективность ваших промптов.

Эта заключительная часть особенно ценна тем, что она обобщает практический опыт и предлагает конкретные стратегии для применения изученных ранее техник. Здесь вы найдете рекомендации по структурированию промптов, работе с JSON и схемами, документированию экспериментов, а также множество нюансов, которые делают разницу между средним и по-настоящему эффективным промптом.

Как и прежде, хочу отметить, что оригинальная публикация фокусируется преимущественно на моделях Gemini и сервисе Vertex AI от Google, однако описанные лучшие практики универсальны и применимы практически ко всем современным большим языковым моделям (GPT, Claude, Llama и др.).

Обратите внимание: Это неофициальный перевод, выполненный энтузиастом с целью сделать ценную информацию доступной русскоязычному сообществу. Некоторые технические термины оставлены без перевода или транслитерированы, поскольку они уже вошли в профессиональный жаргон специалистов в сфере ИИ.

Приятного и полезного чтения!

Лучшие практики

Нахождение правильного промпта требует экспериментов. Language Studio в Vertex AI — идеальное место для работы с вашими промптами, с возможностью тестирования различных моделей.

Используйте следующие лучшие практики, чтобы стать профессионалом в промпт-инжиниринге.

Предоставляйте примеры

Наиболее важная лучшая практика — предоставлять (one shot / few shot) примеры в промпте. Это очень эффективно, поскольку действует как мощный обучающий инструмент. Эти примеры демонстрируют желаемые выводы или похожие ответы, позволяя модели учиться на них и адаптировать собственную генерацию соответственно. Это похоже на предоставление модели ориентира или цели, улучшая точность, стиль и тон ее ответа, чтобы лучше соответствовать вашим ожиданиям.

Проектируйте с простотой

Промпты должны быть краткими, четкими и легкими для понимания как для вас, так и для модели. Как правило, если промпт уже запутан для вас, он, вероятно, будет также запутанным для модели. Старайтесь не использовать сложный язык и не предоставлять ненужную информацию.

ДО:

```
Я сейчас нахожусь в Нью-Йорке и хотел бы узнать больше о замечательных местах. Я с двумя 3-летними детьми. Куда нам следует пойти во время нашего отпуска?
```

ПОСЛЕ ПЕРЕПИСЫВАНИЯ:

```
Выступи в роли туристического гида для туристов. Опиши отличные места для посещения на Манхэттене в Нью-Йорке с 3-летним ребенком.
```

Попробуй использовать глаголы, описывающие действие (прим. переводчика: лучше всего работают глаголы совершенного вида в повелительном наклонении). Вот набор примеров:

Действуй, Проанализируй, Категоризируй, Классифицируй, Сопоставь, Сравни, Создай, Опиши, Определи, Оцени, Извлеки, Найди, Сгенерируй, Идентифицируй, Перечисли, Измерь, Организуй, Разбери, Выбери, Предскажи, Предоставь,

Ранжируй, Рекомендуй, Верни, Извлеки, Перепиши, Выбери, Покажи, Отсортируй, Подытожь, Переведи, Напиши.

Будьте конкретны в отношении вывода

Будьте конкретны в отношении желаемого вывода. Краткая инструкция может недостаточно направить БЯМ или быть слишком общей. Предоставление конкретных деталей в промпте (через системный или контекстуальный промптинг) может помочь модели сфокусироваться на том, что релевантно, улучшая общую точность.

ДЕЛАЙТЕ:

```
Создай блог-пост из 3 абзацев о 5 лучших игровых консолях. Блог-пост должен быть информативным и увлекательным, и он должен быть написан в разговорном стиле.
```

НЕ ДЕЛАЙТЕ:

```
Создай блог-пост об игровых консолях.
```

Используйте инструкции вместо ограничений

Инструкции и ограничения используются в промптинге для направления вывода БЯМ.

- **Инструкция** предоставляет явные указания о желаемом формате, стиле или содержании ответа. Она направляет модель на то, что модель должна делать или производить.
- **Ограничение** — это набор ограничений или границ для ответа. Оно ограничивает то, что модель не должна делать или избегать.

Все больше исследований показывают, что фокус на положительных инструкциях в промптинге может быть более эффективным, чем сильная опора на ограничения. Этот подход соответствует тому, как люди предпочитают положительные инструкции вместо списков того, что не делать.

Инструкции напрямую сообщают желаемый результат, тогда как ограничения могут оставить модель гадать о том, что разрешено. Это дает гибкость и стимулирует творчество в пределах определенных границ, в то время как ограничения могут ограничивать потенциал модели. Также список ограничений может противоречить друг другу.

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

По возможности используйте положительные инструкции: вместо того, чтобы говорить модели, что не делать, скажите ей, что делать вместо этого. Это может избежать путаницы

и улучшить точность вывода.

ДЕЛАЙТЕ:

```
Создай блог-пост из 1 абзаца о 5 лучших игровых консолях. Обсуждай только консоль, компанию, которая её создала, год выпуска и общие продажи.
```

НЕ ДЕЛАЙТЕ:

```
Создай блог-пост из 1 абзаца о 5 лучших игровых консолях. Не перечисляй названия видеоигр.
```

В качестве лучшей практики начинайте с приоритета инструкций, четко указывая, что вы хотите, чтобы модель делала, и используйте ограничения только при необходимости для безопасности, ясности или определенных требований. Экспериментируйте и итерируйте, чтобы протестировать различные комбинации инструкций и ограничений для поиска того, что лучше всего работает для ваших конкретных задач, и документируйте их.

Контролируйте максимальную длину токенов

Для контроля длины сгенерированного ответа БЯМ вы можете либо установить ограничение максимального количества токенов в конфигурации, либо явно запросить определенную длину в вашем промпте. Например:

```
"Объясни квантовую физику в сообщении длиной с твит."
```

Используйте переменные в промптах

Для повторного использования промптов и сделать их более динамичными используйте переменные в промпте, которые могут быть изменены для разных входных данных. Например, как показано в Таблице 20, промпт, который даёт факты о городе. Вместо жесткого кодирования названия города в промпте, используйте переменную. Переменные могут сэкономить вам время и усилия, позволяя избежать повторений. Если вам нужно использовать один и тот же кусок информации в нескольких промптах, вы можете сохранить его в переменной, а затем ссылаться на эту переменную в каждом промпте. Это имеет особый смысл при интеграции промптов в ваши собственные приложения.

Промпт	ПЕРЕМЕННЫЕ {city} = "Амстердам" ПРОМПТ Ты туристический гид. Расскажи мне факт о городе: {city}
Вывод	Амстердам — красивый город, полный каналов, мостов и узких улочек. Это отличное место для посещения благодаря его богатой истории, культуре и ночной жизни.

Таблица 20. Использование переменных в промптах

Экспериментируйте с форматами ввода и стилями письма

Разные модели, конфигурации моделей, форматы промптов, выбор слов и подходы могут давать разные результаты. Поэтому важно экспериментировать с атрибутами промпта, такими как стиль, выбор слов и тип промпта (с нулевым примером, с несколькими примерами, системный промпт).

Например, промпт с целью генерации текста о революционной игровой консоли Sega Dreamcast может быть сформулирован как **вопрос**, **утверждение** или **инструкция**, в результате чего получаются разные выводы:

- **Вопрос**: Что представляла собой Sega Dreamcast и почему она была такой революционной консолью?
- **Утверждение**: Sega Dreamcast была игровой консолью шестого поколения, выпущенной Sega в 1999 году. Она...
- **Инструкция**: Напишите один абзац, который описывает консоль Sega Dreamcast и объясняет, почему она была такой революционной.

Для промптинга с несколькими примерами при задачах классификации перемешивайте классы

В общем случае порядок примеров с несколькими примерами не должен сильно влиять. Однако при выполнении задач классификации обязательно перемешивайте возможные классы ответов в примерах с несколькими примерами. Это потому, что иначе вы можете переобучиться на конкретный порядок примеров. Перемешивая возможные классы ответов, вы можете убедиться, что модель учится идентифицировать ключевые особенности каждого класса, а не просто запоминать порядок примеров. Это приведет к более надежной и обобщаемой производительности на несиденных данных.

Хорошее эмпирическое правило — начать с 6 примеров для few-shot промптинга и проверять точность с этого момента.

Адаптируйтесь к обновлениям моделей

Важно быть в курсе изменений архитектуры модели, добавленных данных и возможностей. Попробуйте новые версии моделей и адаптируйте свои промпты, чтобы лучше использовать новые функции модели. Инструменты вроде Vertex AI Studio отлично подходят для хранения, тестирования и документирования различных версий вашего промпта.

Экспериментируйте с форматами вывода

Помимо формата ввода промпта, рассмотрите возможность экспериментов с форматом вывода. Для нетворческих задач, таких как извлечение, выбор, разбор, упорядочивание, ранжирование или категоризация данных, попробуйте получить вывод в структурированном формате, таком как JSON или XML.

Есть некоторые преимущества в возврате JSON-объектов из промпта, который извлекает данные. 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, это заставляет модель создать структуру и ограничить галлюцинации.

В общем, преимущества использования JSON для вашего вывода:

- Всегда возвращается в одном и том же стиле
- Фокус на данных, которые вы хотите получить
- Меньше шансов на галлюцинации
- Осведомленность о взаимосвязях
- Вы получаете типы данных
- Вы можете сортировать данные

Таблица 4 в разделе о промптинге с несколькими примерами показывает пример того, как вернуть структурированный вывод.

Восстановление JSON

Хотя возврат данных в формате JSON предлагает множество преимуществ, это не лишено недостатков. Структурированная природа JSON, хотя и полезна для разбора и использования в приложениях, требует значительно больше токенов, чем обычный текст, что приводит к увеличению времени обработки и более высоким затратам. Кроме того, многословность JSON может легко потреблять все окно вывода, что становится особенно проблематичным, когда генерация внезапно обрывается из-за ограничений токенов. Это усечение часто приводит к недопустимому JSON, отсутствию важных закрывающих фигурных скобок или квадратных скобок, что делает вывод непригодным для использования. К счастью, такие инструменты, как библиотека json-repair (доступная на PyPI), могут быть неоценимы в этих ситуациях. Эта библиотека интеллектуально пытается автоматически исправлять неполные или неправильно сформированные JSON-объекты, что делает ее важным союзником при работе с JSON, сгенерированным БЯМ, особенно при работе с потенциальными проблемами усечения.

Работа со схемами

Использование структурированного JSON в качестве вывода – отличное решение, как мы видели несколько раз в этой статье. Но как насчет *ввода*? Хотя JSON отлично подходит для структурирования *вывода*, генерируемого БЯМ, он также может быть невероятно полезен для структурирования *ввода*, который вы предоставляете. Здесь в игру вступают JSON-схемы. JSON-схема определяет ожидаемую структуру и типы данных вашего JSON-ввода. Предоставляя схему, вы даёте БЯМ четкий план данных, которые она должна ожидать,

помогая ей сосредоточить своё внимание на соответствующей информации и снижая риск неправильного толкования ввода. Более того, схемы могут помочь установить отношения между различными частями данных и даже сделать БЯМ "осведомленной о времени", включая поля даты или временной метки с определенными форматами.

Вот простой пример:

Допустим, вы хотите использовать БЯМ для генерации описаний продуктов в каталоге электронной коммерции. Вместо того, чтобы просто предоставлять описание продукта в свободной форме, вы можете использовать JSON-схему для определения атрибутов продукта:

```
{
 "type": "object",
 "properties": {
 "name": { "type": "string", "description": "Название продукта" },
 "category": { "type": "string", "description": "Категория продукта" },
 "price": { "type": "number", "format": "float", "description": "Цена продукта" },
 "features": {
 "type": "array",
 "items": { "type": "string" },
 "description": "Ключевые особенности продукта"
 },
 "release_date": { "type": "string", "format": "date", "description": "Дата выпуска продукта"}
 },
```

Фрагмент кода 5. Определение схемы структурированного вывода

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

```
{
 "name": "Беспроводные наушники",
 "category": "Электроника",
 "price": 99.99,
 "features": ["Шумоподавление", "Bluetooth 5.0", "20-часовой срок службы батареи"],
 "release_date": "2023-10-27"
}
```

Фрагмент кода 6. Структурированный вывод от БЯМ

Предварительно обрабатывая ваши данные и вместо предоставления полных документов, предоставляя как схему, так и данные, вы даёте БЯМ четкое понимание атрибутов

продукта, включая дату его выпуска, что делает гораздо более вероятным генерацию точного и релевантного описания. Этот структурированный подход к вводу, направляющий внимание БЯМ на соответствующие поля, особенно ценен при работе с большими объемами данных или при интеграции БЯМ в сложные приложения.

Экспериментируйте вместе с другими промпт-инженерами

Если вы находитесь в ситуации, когда вам нужно попытаться придумать хороший промпт, вы можете захотеть найти нескольких людей для попытки. Когда все следуют лучшим практикам (как перечислено в этой главе), вы увидите разницу в производительности между всеми различными попытками промптов.

Лучшие практики CoT

При использовании промптинга с цепочкой рассуждений (CoT) размещение ответа после рассуждения является обязательным, поскольку генерация рассуждения изменяет токены, которые модель получает при прогнозировании окончательного ответа.

При работе с CoT и самосогласованностью вам необходимо иметь возможность извлекать окончательный ответ из вашего промпта отдельно от рассуждения.

Для промптинга CoT установите температуру на 0.

Промптинг с цепочкой рассуждений основан на жадном декодировании, предсказывающем следующее слово в последовательности на основе наивысшей вероятности, присвоенной языковой моделью. В общем случае, когда используется рассуждение для получения окончательного ответа, скорее всего, существует единственный правильный ответ. Поэтому температура всегда должна быть установлена на 0.

Документируйте различные попытки промптов

Последний совет уже упоминался в этой главе, но мы не можем не подчеркнуть, насколько это важно: документируйте свои попытки промптов во всех деталях, чтобы вы могли со временем узнать, что сработало хорошо, а что нет.

Вывод промптов может различаться между моделями, настройками сэмплирования и даже между разными версиями одной и той же модели. Более того, даже для идентичных промптов к одной и той же модели могут возникать небольшие различия в форматировании предложений вывода и выборе слов. (Например, как упоминалось ранее, если два токена имеют одинаковую предсказанную вероятность, связи могут быть разорваны случайным образом. Это может затем повлиять на последующие предсказанные токены.).

Мы рекомендуем создать Google Таблицу с Таблицей 21 в качестве шаблона. Преимущества такого подхода заключаются в том, что у вас есть полная запись, когда вы неизбежно

должны будете вернуться к своей работе по промптингу – либо чтобы продолжить ее в будущем (вы бы удивились, как много можно забыть даже после короткого перерыва), либо для проверки производительности промпта на разных версиях модели, а также для помощи в отладке будущих ошибок.

Помимо полей в этой таблице, также полезно отслеживать версию промпта (итерацию), поле для фиксации, был ли результат OK/NOT OK/SOMETIMES OK, и поле для фиксации обратной связи. Если вам посчастливилось использовать Vertex AI Studio, сохраняйте свои промпты (используя то же имя и версию, что и указаны в вашей документации) и отслеживайте гиперссылку на сохраненный промпт в таблице. Таким образом, вы всегда находитесь в одном клике от повторного запуска ваших промптов.

При работе с системой извлечения, дополненной генерацией (*retrieval augmented generation*), вы также должны фиксировать конкретные аспекты системы RAG, которые влияют на то, какой контент был вставлен в промпт, включая запрос, настройки чанкинга, вывод чанкинга и другую информацию.

Как только вы почувствуете, что промпт близок к совершенству, перенесите его в вашу кодовую базу проекта. И в кодовой базе сохраняйте промпты в отдельном файле от кода, чтобы их было легче поддерживать. Наконец, в идеале ваши промпты являются частью операционализированной системы, и как промпт-инженер вы должны полагаться на автоматизированные тесты и процедуры оценки, чтобы понимать, насколько хорошо ваш промпт обобщается для задачи.

Промпт-инжиниринг – это итеративный процесс. Создавайте и тестируйте различные промпты, анализируйте и документируйте результаты. Улучшайте свой промпт на основе производительности модели. Продолжайте экспериментировать, пока не достигнете желаемого вывода. Когда вы меняете модель или конфигурацию модели, вернитесь и продолжайте экспериментировать с ранее использованными промптами.

Название	[имя и версия вашего промпта]
Цель	[Объяснение цели этой попытки в одном предложении]
Модель	[название и версия используемой модели]
Температура	[значение между 0 - 1]
Top-K	[число]
Промпт	[Запишите весь полный промпт]
Вывод	[Запишите вывод или несколько выводов]

Таблица 21. Шаблон для документирования промптов

Резюме

В этой статье обсуждается промпт-инжиниринг. Мы изучили различные техники промптинга, такие как:

- Промптинг с нулевым примером
- Промптинг с несколькими примерами
- Системный промптинг
- Ролевой промптинг
- Контекстуальный промптинг
- Промптинг с отступлением
- Цепочка рассуждений
- Самосогласованность
- Дерево рассуждений
- ReAct

Мы даже рассмотрели способы автоматизации ваших промптов.

Затем в статье обсуждаются проблемы генеративного ИИ, такие как проблемы, которые могут возникнуть, когда ваши промпты недостаточны. Мы завершили лучшими практиками о том, как стать лучшим промпт-инженером.

Завершение цикла

На этом мы завершаем наш цикл переводов о промпт-инжиниринге. За три части мы прошли путь от базовых концепций до продвинутых техник и лучших практик, формируя целостное представление об искусстве и науке эффективного взаимодействия с большими языковыми моделями.

Мы познакомились с фундаментальными техниками промптинга, изучили сложные методы вроде цепочки рассуждений и промптинга с отступлением, а также разобрали практические рекомендации по оптимизации промптов для решения конкретных задач.

Эта область знаний продолжает активно развиваться, и каждый день появляются новые подходы и методики. Промпт-инжиниринг находится на пересечении искусства и технологии, сочетая творческий подход к формулировке задач и техническое понимание принципов работы языковых моделей.

Призыв к обмену опытом!

Я заметил, что, несмотря на значительное количество просмотров и добавлений в закладки предыдущих частей цикла, комментариев было относительно немного. Хотелось бы изменить эту ситуацию!

Какие техники промптинга вы применяете в своей работе? С какими сложностями сталкиваетесь? Возможно, у вас есть собственные оригинальные подходы или модификации описанных методов? Какие результаты вы получили, применяя изученные техники на практике?

Поделитесь своими находками в комментариях — это бесценно для всего сообщества! Практический опыт реальных пользователей часто оказывается не менее важным, чем теоретические разработки.

Также интересно было бы узнать, какие темы, связанные с промпт-инжинирингом и работой с БЯМ, вы хотели бы увидеть в будущих публикациях. Многие аспекты применения ИИ остаются за рамками даже такого подробного материала, как этот.

Благодарю вас за внимание к этому циклу статей, и надеюсь, что этот материал станет полезным подспорьем в вашей работе с большими языковыми моделями!

Revision #2
Created 2026-02-16 09:03:54 UTC by Эд Кукса
Updated 2026-02-16 09:09:25 UTC by Эд Кукса