Ростелеком

3.1.2 Интеграция API ИИ

Модуль 3. Интеграция приложений и prompt engineering

Оглавление

1. Введение

Сегодня мы с вами продолжаем разговор о важной и прикладной теме — интеграции больших языковых моделей с веб-приложениями. Мы разберем, как использовать возможности LLM не просто в режиме чата, а как полноценный элемент вашего цифрового продукта, будь то сайт, бот, внутренняя система или пользовательское веб-приложение.

Когда мы говорим об интеграции с API в контексте Prompt Engineering, мы имеем в виду возможность управлять работой модели не вручную, а программно — через код. API (Application Programming Interface), или программный интерфейс, позволяет отправлять запросы к модели, получать ответы и включать эту коммуникацию в логику приложения. Это уже не просто интерактивная сессия, где вы сидите перед окном чата, это автоматизация: вы подаете запросы с помощью кода, контролируете параметры модели, и результат работы используется дальше в программе.

Давайте еще раз уточним, чем отличаются сценарии просто писать промпты от написания промптов для API. В обычной ситуации вы формулируете запрос, исходя из текущего контекста и решаете конкретную задачу, думая как собеседник умного ассистента. Например, просите «Составь план путешествия по золотому кольцу России».

Но когда вы работаете с API, вы уже проектируете промпт, который будет включать в себя запрос, который приходит от других пользователей и заранее неизвестен. Суммарный промпт должен быть предсказуемым, устойчивым к изменениям данных и работать в разных ситуациях. Вам важно учитывать, как промпт будет формироваться в

коде, как он будет адаптироваться под входные данные, и как обеспечить стабильное поведение модели. Это уже не творчество ради ответа, это инженерная задача — сделать промпт частью программного решения.

Сегодня мы рассмотрим, как всё это реализуется на практике — как строятся запросы к LLM, и как связать все это с пользовательским интерфейсом или бэкендом вашего приложения.

2. Методы запросов GET

Когда вы начинаете работать с API, особенно при интеграции языковых моделей в веб-приложения, вы сталкиваетесь с такими понятиями, как GET и POST-запросы. Это два основных способа, с помощью которых ваше приложение может «общаться» с внешним сервисом, например, с LLM через API.

Запросы могут выполняться как на сервере, так и прямо в браузере клиента, вариантов множество. Рассмотрим пример запроса на JavaScript (Fetch API).

GET-запрос используется, когда вам нужно просто получить какую-то информацию. Браузер выполняет javaScript код и делает GET-запрос, чтобы получить данные и показать их вам. В случае работы с LLM это может быть не так часто применимо, потому что такие запросы обычно не передают большие объемы текста и требуют секретных ключей. Но они все же могут использоваться, например, для получения информации о доступных моделях, лимитах или базовой информации.

3. Методы запросов POST

POST-запрос — это то, что применяется чаще всего при работе с языковыми моделями. Он используется, когда вы отправляете данные на сервер. То есть вы формируете запрос, вкладываете в него промпт, параметры модели, возможно, какие-то дополнительные настройки, и отправляете это на сервер — модель анализирует и возвращает ответ. Важно понимать, что POST-запрос — это как письмо с вложением: вы передаёте что-то внутрь, и получаете результат обработки.

С технической стороны GET и POST отличаются тем, как передаются данные. В GET всё передаётся прямо в адресе запроса, а в POST данные отправляются отдельно, «в теле» запроса. Это делает POST удобнее и безопаснее для передачи текстов, особенно длинных промптов, инструкций или пользовательских данных.

По сути, когда вы пишете промпт и хотите, чтобы ваш код передал его в модель — вы чаще всего делаете это через POST. И именно с такими запросами мы будем работать, когда начнем подключать LLM к веб-приложениям.

4. Заголовки и тело запроса

Когда вы отправляете запрос к API языковой модели, важно понимать, из чего этот запрос состоит. По сути, каждый запрос — это как письмо: у него есть адрес, заголовки и тело. Адрес указывает, куда именно вы обращаетесь — это URL, по которому находится нужная вам функция модели. Он может включать параметры, если это GET-запрос, или просто быть указанием на конечную точку, если вы отправляете POST.

Дальше идут заголовки. Заголовки — это служебная информация, которая говорит серверу, как обрабатывать ваш запрос. Один из самых важных заголовков — это тот, который передаёт ключ доступа к API, то есть ваш персональный идентификатор. Без него большинство сервисов просто не примут ваш запрос. Также заголовки сообщают, в каком формате вы отправляете данные и в каком формате ждёте ответ. Например, чаще всего используется формат JSON — это текст, оформленный по определённым правилам, понятным и серверу, и вашему приложению.

Основная часть запроса — это тело. Именно в теле передаётся содержимое промпта и все параметры, которые вы хотите задать модели: температура, максимальная длина ответа, системные инструкции и так далее. Вы можете представить это как содержимое формы, которую вы заполняете, чтобы отправить модель за результатом. То, что вы вкладываете в тело, во многом определяет, как модель будет себя вести и что именно она вам вернёт.

Таким образом, когда вы создаёте запрос из своего веб-приложения, вы формируете адрес, настраиваете заголовки и заполняете тело — и всё это вместе отправляется модели. Ваша задача как инженера — сделать этот запрос точным, понятным и устойчивым, чтобы модель отвечала так, как вы ожидаете.

5. Что такое REST API

REST API — это один из самых популярных способов, с помощью которых приложения могут обмениваться данными через интернет. Он построен на простых принципах: вы обращаетесь к определённым

адресам, отправляете запросы, и получаете ответы, чаще всего в формате JSON. REST — это архитектурный стиль, в котором всё выглядит и работает как обычный веб: есть адреса, есть методы, вроде GET или POST, и есть данные, которые передаются туда и обратно. Когда вы интегрируете LLM через REST API, вы просто отправляете HTTP-запрос с нужной информацией и получаете от модели сгенерированный ответ.

Обработка запроса – это не бесплатное мероприятие, поэтому почти всегда нужен API-ключ. Это как личный пропуск. Он говорит серверу, кто вы, имеет ли ваше приложение доступ, и что именно вам разрешено делать. Ключ нельзя публиковать в открытом доступе, особенно на клиентской стороне, потому что он даёт возможность пользоваться ресурсами, за которые вы платите. Обычно этот ключ передаётся в заголовках запроса, и без него сервис просто не ответит.

Ограничения по токенам — это техническая особенность языковых моделей. Модель не работает с буквами или словами напрямую, она оперирует токенами — это такие маленькие единицы текста, которыми она разбивает любую фразу. В каждом запросе и ответе считается общее количество токенов, и у каждой модели есть свой предел. Например, если лимит 4000 токенов, это означает, что сумма вашего промпта и ответа модели не может превысить этот предел. Если вы превысите лимит, запрос может не выполниться или часть ответа будет обрезана.

Платность и ограничения по количеству запросов — это способ контролировать нагрузку на сервис и обеспечить его стабильную работу. Обычно API имеет тарифы: чем больше вы используете, тем

больше платите. Также может быть ограничение на количество запросов в минуту или в сутки, чтобы никто не мог перегрузить сервер. Это называется rate limit — ограничение скорости. Если вы превысите лимит, вы получите ошибку, и запрос придётся повторить позже. Эти ограничения стоит учитывать, особенно если вы строите продукт с большим числом пользователей или с интенсивным использованием модели.

6. Формат JSON

Формат JSON — это способ представления данных в виде текста, который понятен и человеку, и компьютеру. Он очень широко используется в веб-разработке и особенно важен при работе с API, в том числе с языковыми моделями. JSON напоминает обычные словари или объекты: вы описываете данные как пары «ключ — значение». Для обозначения объекта используются фигурные скобки. Ключом может быть, например, слово prompt, а значением — текст вашего запроса к модели. Такие пары объединяются в структуру, которая может включать внутри себя другие данные: списки, числа, текст, логические значения. Для массивов из нескольких объектов используют квадратные скобки.

Почему JSON удобен? Потому что он легко читается, его просто создавать программно, и большинство языков программирования умеют с ним работать без дополнительной настройки. Когда вы отправляете POST-запрос к LLM, вы вкладываете всё содержимое — и сам промпт, и параметры вроде максимального числа токенов, и настройки генерации — именно в формате JSON. Сервер читает этот

текст, разбирает его структуру и понимает, что вы хотите от модели. Также и ответ от модели приходит к вам в виде JSON, где вы можете найти нужный текст, например, по ключу content или message.

Важно помнить, что JSON — это не просто текст, а строго структурированный текст. Все кавычки, запятые и скобки имеют значение. Если структура нарушена, сервер не сможет прочитать ваш запрос. Поэтому, когда вы формируете JSON в коде, нужно быть особенно внимательным к его синтаксису. Если всё сделано правильно, обмен данными с моделью становится надёжным, понятным и удобным.

7. Ключ безопасности API LLM

Ключ безопасности API — это специальный код, который позволяют вашему приложению пользоваться возможностями внешнего сервиса, например, языковой модели. Ключи выполняют сразу несколько функций. Прежде всего, они подтверждают вашу личность — сервер понимает, кто именно делает запрос. Кроме того, по этому ключу можно отслеживать, сколько запросов вы отправили, какие функции вы используете и сколько ресурсов потребляете. Это важно, потому что работа с LLM требует серьёзных вычислительных мощностей, и такие ресурсы стоят дорого.

Каждый раз, когда вы отправляете запрос к модели, запускается сложная система, которая обрабатывает ваш промпт, генерирует ответ и возвращает его обратно. За этим стоит работа большого количества процессоров и видеокарт, которые находятся в дата-центрах. Чем больше размер модели и сложнее запрос, тем выше стоимость этого

процесса. Поэтому запуск LLM — это не просто получение текста, это вычисление, за которое платят, как правило, по количеству использованных токенов.

Именно из-за этого важно ограничивать доступ к API и защищать свои ключи. Если ваш ключ попадёт в чужие руки, кто-то может начать отправлять большое количество запросов от вашего имени, и вы просто получите огромный счёт. Также это может привести к перегрузке вашего приложения или к временной блокировке из-за превышения лимитов. Поэтому всегда храните API-ключи в защищённом виде, не вставляйте их напрямую в клиентский код, и используйте специальные методы для контроля доступа. Это основа безопасной и экономически разумной работы с языковыми моделями в ваших веб-приложениях.

8. Параметры генерации

Когда вы отправляете запрос к языковой модели через API, у вас есть возможность управлять её поведением с помощью параметров. Это позволяет настраивать стиль, длину, предсказуемость и даже содержание ответа. Один из самых важных параметров — это temperature. Он определяет степень случайности в генерации текста. Если вы задаёте низкое значение, например 0.2, модель будет отвечать очень точно, выбирая самые вероятные слова. Если вы увеличиваете его до 0.8 или выше, в ответах появляется больше разнообразия и креативности, но они становятся менее предсказуемыми.

Похожий параметр — top_p. Он управляет тем, насколько широко модель смотрит на возможные варианты ответа. Если задать top_p

равным 1, модель учитывает все возможные продолжения. Если указать 0.5, она будет выбирать только из самых вероятных слов, отбрасывая редкие. Оба параметра отвечают за баланс между стабильностью и оригинальностью, и обычно используют либо один, либо другой.

Параметр max_tokens позволяет ограничить длину ответа. Это полезно, если вам важно, чтобы модель не выходила за рамки, особенно в условиях жёстких ограничений по количеству токенов или если вы хотите получить компактный результат. Если вы не укажете этот параметр, модель может сгенерировать слишком длинный текст или, наоборот, слишком короткий.

Параметр stop позволяет задать сигналы, по которым модель должна прекратить генерацию текста. Например, если вы ожидаете, что ответ должен остановиться при появлении определённого слова или символа, вы указываете это значение в параметре stop. Это помогает контролировать структуру ответа, особенно в автоматических системах, где важно точно отделять один блок текста от другого.

Есть ещё два параметра, которые тонко управляют тем, как модель выбирает слова — frequency_penalty и presence_penalty. Первый уменьшает вероятность повторов, если слово уже встречалось часто. Второй снижает шансы повторного использования слов, которые уже были упомянуты, даже если только один раз. Оба параметра помогают сделать текст менее однообразным, но с разным эффектом: один борется с частыми повторами, другой — с простым повторением фактов. Настраивая эти параметры, вы можете добиться либо более

сдержанных и логичных ответов, либо более разнообразных и живых, в зависимости от цели вашего приложения.

9. Пример ответа

Когда вы отправляете запрос к LLM через API, вы получаете в ответ структуру в формате JSON. Давайте посмотрим на упрощённый пример и разберём, что означает каждое поле.

Поле `id` — это уникальный идентификатор конкретного запроса. Оно нужно, чтобы вы могли отследить или зафиксировать каждый индивидуальный запрос и разобраться, если что-то пойдет не так.

Поле `object` указывает тип возвращаемого объекта. В случае ChatGPT это `chat.completion`, что говорит о том, что вы получили результат диалога, а не, например, ошибку или другую структуру.

Поле `created` — это отметка времени, когда был сгенерирован ответ. Представлено в формате Unix времени — это количество секунд с 1 января 1970 года. Обычно его используют для логирования и аналитики.

Поле `model` показывает, какая именно модель использовалась для генерации — например, `gpt-4`. Это важно, если вы работаете с несколькими версиями и хотите сравнивать поведение.

Поле `choices` — это список возможных ответов от модели. Обычно там один вариант, но можно запросить несколько. Внутри каждого элемента есть `index`, который указывает номер варианта, `message`, в котором хранится сам ответ и роль, с которой модель говорит. В данном случае роль — `assistant`, а `content` содержит сгенерированный текст. Поле `finish_reason` говорит, почему модель

остановила генерацию. Значение `stop` означает, что ответ завершён нормально. Бывают другие причины, например, достижение лимита токенов.

Наконец, блок `usage` содержит информацию о том, сколько токенов было использовано. `prompt_tokens` — это токены, которые вы отправили в запросе. `completion_tokens` — это токены, сгенерированные моделью в ответе. `total_tokens` — это их сумма. Эти данные нужны вам, чтобы отслеживать расход, так как стоимость использования модели рассчитывается по количеству токенов.

Понимание этой структуры помогает вам правильно обрабатывать ответы от модели в коде, извлекать нужные данные и учитывать расход при масштабировании.

10. Роль промпт-инженера в команде

Роль промпт-инженера в команде веб-разработки — это связующее звено между возможностями языковой модели и потребностями продукта. Когда команда создает веб-приложение с интеграцией ИИ, задача промпт-инженера — сделать так, чтобы модель отвечала именно так, как нужно пользователю, с нужным тоном, структурой и содержанием. Это не просто написать хороший текст запроса, а выстроить систему взаимодействия с моделью: понять, какую информацию ей нужно давать, как её формулировать и как обрабатывать ответ.

Вы работаете с разработчиками и продакт-менеджерами, чтобы понять, какую функцию в продукте должен выполнять ИИ. Затем вы описываете поведение модели с помощью промптов. Вы можете

создавать шаблоны запросов, где часть текста меняется в зависимости от пользовательского ввода, а часть задаёт стиль и правила общения. Например, вы можете указать, что модель должна говорить строго, вежливо, на деловом языке и не использовать эмодзи — и всё это включается в системное сообщение или в начало запроса.

Также вы тестируете, как модель реагирует на разные входные данные, проверяете стабильность и надёжность генерации, подбираете параметры вроде температуры или максимального количества токенов. Иногда вы проектируете целые сценарии, где модель ведёт диалог или выполняет несколько шагов подряд. Чем точнее вы опишете поведение модели, тем проще разработчикам встроить её в интерфейс и тем выше будет качество пользовательского опыта.

Представим, что вы работаете над веб-приложением, которое помогает пользователям переписывать деловые письма в более вежливом и профессиональном тоне одним нажатием кнопки. Ваша задача как промпт-инженера — создать такой промпт, чтобы модель стабильно выполняла эту функцию.

Важно, что вы заранее проговариваете с разработчиками, как этот промпт будет встроен: через системное сообщение или как часть пользовательского ввода, какие параметры модели использовать, сколько токенов выделить, что делать, если письмо слишком длинное. Вы тестируете результат, уточняете формулировки, добиваетесь предсказуемости и качества.

Хороший промпт — это мост между логикой продукта и поведением модели. Именно вы как промпт-инженер отвечаете за то,

чтобы модель встраивалась в продукт не как универсальный чат, а как целенаправленный инструмент, обученный выполнять конкретную задачу. Это требует не только понимания языка, но и глубокого понимания логики продукта, целей пользователей и ограничений, связанных с API.

11. Заключение

Сегодня мы с вами рассмотрели ключевые аспекты интеграции API искусственного интеллекта, которые являются фундаментом для успешной работы в области промпт-инжиниринга. Давайте кратко подведем итоги нашего обсуждения.

Мы изучили основные методы взаимодействия с API через GET и POST запросы, разобрали структуру HTTP-заголовков и тела запроса. Понимание этих механизмов позволяет вам эффективно формировать запросы к языковым моделям и получать желаемые результаты.

REST API, как архитектурный стиль, предоставляет нам удобный способ организации взаимодействия с ИИ-сервисами. Формат JSON, который мы рассмотрели, стал стандартом для передачи данных, обеспечивая простоту и универсальность в работе.

Вопросы безопасности API и работы с ключами доступа – это критически важный аспект, который необходимо всегда держать в фокусе внимания. Правильная настройка параметров генерации позволяет контролировать поведение модели и получать более точные ответы.

Особую роль в современном IT-пространстве играет промпт-инженер, который выступает связующим звеном между бизнес-задачами и возможностями ИИ-технологий. Помните, что успешная интеграция ИИ-решений требует комплексного подхода: технических знаний, понимания бизнес-процессов и умения работать в команде. Каждый из рассмотренных сегодня аспектов играет важную роль в создании эффективных решений с использованием возможностей ИИ.

Благодарю за внимание!