Ростелеком

3.3.1 Технологии мобильных приложений

Модуль 3. Интеграция приложений и prompt engineering

Оглавление

1. Введение

Здравствуйте, уважаемые коллеги!

Сегодня мы с вами погрузимся в вопросы интеграции искусственного интеллекта в мобильные приложения – одну из самых перспективных областей современной разработки. Эта тема особенно актуальна в условиях стремительного развития технологий и растущего спроса на интеллектуальные мобильные решения.

Мобильные приложения давно перестали быть просто инструментами для коммуникации. Сегодня это полноценные платформы, которые помогают решать сложные задачи, автоматизировать бизнес-процессы и улучшать качество жизни пользователей. Интеграция ИИ открывает новые возможности для создания инновационных продуктов, способных адаптироваться к потребностям пользователей и предоставлять персонализированный опыт.

В ходе нашей лекции мы рассмотрим ключевые аспекты взаимодействия **prompt инженера** с мобильной разработкой. Вы узнаете о различных подходах к созданию мобильных приложений: от нативных решений до кроссплатформенных фреймворков. Особое внимание уделим архитектуре **REST API** и работе с локальными языковыми моделями.

Мы разберем практические примеры использования ИИ в мобильном приложении и подробно рассмотрим создание бота на платформе Telegram. Вы узнаете о роли промптов в работе таких систем и освоите принципы построения архитектуры ботов.

Понимание этих технологий позволит вам успешно реализовывать проекты в области AI-разработки. Полученные знания станут важным шагом в освоении профессии промпт-инженера и помогут вам быть востребованным специалистом на рынке труда.

2. Prompt engineering и мобильная разработка

Prompt engineering и мобильная разработка тесно связаны, потому что сегодня мобильные приложения всё чаще используют возможности искусственного интеллекта и языковых моделей. Вы, как специалисты по созданию запросов к ИИ, фактически становитесь мостом между пользователем и умным алгоритмом, который работает внутри приложения. Представьте, что внутри мобильного сервиса, например, чат-бота или помощника по здоровью, есть нейросеть вроде ChatGPT. Но чтобы она работала корректно, с нужной логикой, тоном, реакцией на ошибки, — ей нужно правильно сформулировать запрос. Именно это делает prompt инженер. Он придумывает, как и что сказать модели, чтобы она выдала нужный результат в нужной ситуации.

Если пользователь взаимодействует с ИИ через мобильное приложение он не видит самих запросов. Зато вы их проектируете — заранее, на все возможные случаи. Вы думаете, что должен сказать голосовой помощник, когда пользователь жалуется на плохую погоду, как объяснить сложную настройку, как корректно реагировать на грубость. И всё это — через promt-логику, вплетённую в код приложения.

Таким образом, ваша задача — создавать такие текстовые сценарии взаимодействия, которые легко внедряются в мобильное приложение и делают поведение ИИ понятным, полезным и человечным. Вы работаете на стыке технологий и коммуникации, а мобильная среда — это просто одна из самых востребованных площадок для реализации вашего мастерства.

3. Нативные приложения

Нативные приложения — это такие программы, которые создаются специально для конкретной операционной системы, например для iOS от Apple или Android от Google. Это значит, что приложение для iPhone пишется с учётом всех особенностей iOS, а для Android — с учётом всех правил и возможностей Android-системы. Разработчики используют официальные языки программирования и инструменты. Для iOS это, как правило, язык Swift и среда Xcode. Для Android — язык Kotlin или Java и среда Android Studio.

Такие приложения устанавливаются прямо на устройство через App Store или Google Play. Они имеют прямой доступ к функциям телефона — камере, микрофону, геолокации, сенсорам, Bluetooth. Поэтому работают быстро и плавно. Визуально они тоже чувствуют себя как дома, потому что следуют стилю операционной системы. Всё выглядит и ощущается так, будто сделано именно для этого устройства.

Для вас, как для prompt инженеров, важно понимать, что нативное приложение может включать в себя встроенные модули, которые используют ИИ. Это может быть чат, генератор текста, голосовой помощник. И здесь вы отвечаете за то, как будет организовано

взаимодействие между пользователем и ИИ: что именно будет передаваться в модель, как будут обрабатываться ответы, как пользовательский интерфейс будет реагировать на разные сценарии. Вы работаете вместе с мобильными разработчиками, чтобы ваши запросы к ИИ были правильно встроены в логику приложения, и чтобы весь опыт работы с ИИ ощущался естественным, быстрым и качественным.

4. Кроссплатформенные фреймворки

Кроссплатформенные фреймворки — это инструменты, с помощью которых можно создать одно приложение, которое будет работать и на iOS, и на Android. Вместо того чтобы писать два отдельных приложения для каждой платформы, разработчики пишут один общий код, и этот код потом превращается в программу, которая запускается на обоих типах устройств. Это экономит время, деньги и усилия.

Один из популярных фреймворков — Flutter от Google. В нём используется язык программирования Dart. Когда вы создаёте интерфейс во Flutter, он одинаково выглядит и работает на разных устройствах, потому что Flutter сам отрисовывает элементы на экране, независимо от того, какая система внутри телефона. React Native — это ещё один кроссплатформенный инструмент. Он использует JavaScript и позволяет собирать интерфейс из компонентов, которые взаимодействуют с системными элементами iOS и Android.

Для вас как для prompt инженеров это важно, потому что ваше взаимодействие с ИИ должно быть универсальным и стабильным на разных устройствах. Если вы создаёте запрос к модели, он должен

одинаково работать и на айфоне, и на андроиде, независимо от того, каким фреймворком написано приложение. Вы также должны учитывать, как быстро и удобно приложение будет общаться с моделью, передавать данные, получать ответы и отображать их. Кроссплатформенные технологии позволяют быстрее внедрять и тестировать такие функции, а значит, вы быстрее проверяете эффективность ваших промтов и улучшаете пользовательский опыт.

5. Архитектура REST API

Когда вы используете LLM в мобильном приложении, чаще всего взаимодействие с ней происходит через API по протоколу REST. Архитектура приложения в этом случае устроена так: на вашем устройстве — телефоне или планшете — работает клиентская часть, то есть интерфейс, с которым взаимодействует пользователь. Он вводит текст, нажимает кнопку, говорит голосом — всё это собирается в запрос.

Затем этот запрос отправляется на сервер, где расположен API языковой модели. Это может быть облачный сервис, например от OpenAI. Передача данных происходит через интернет, чаще всего в формате JSON, по протоколу HTTPS. Такой запрос содержит ваш prompt, параметры, которые вы заранее заложили, и токен авторизации. Сервер обрабатывает запрос, модель генерирует ответ, и этот ответ возвращается обратно на устройство.

Мобильное приложение получает результат и отображает его в нужном виде — как текст, как карточку, как сообщение в чате. И тут важна ваша роль как prompt инженера. Именно вы определяете, как

будет выглядеть запрос к модели, как структурировать данные, чтобы получить правильный и полезный ответ, и как обрабатывать ответы от модели. Вы можете заранее продумать, как приложение должно реагировать на разные типы ответов, как вести себя в случае ошибок, и как сделать всё так, чтобы пользователь чувствовал, будто общается с умным, понятным и отзывчивым собеседником.

Вы работаете вместе с разработчиками, чтобы встроить ваши промты в эту архитектуру. Вы создаёте шаблоны, управляете переменными в запросе, следите за качеством диалогов. REST API — это просто канал связи, а вы — тот, кто формулирует мысли, которые через этот канал передаются в модель и обратно.

6. Локальные LLM

Когда вы используете локальную LLM, такую как llama.cpp, архитектура приложения меняется по сравнению с облачной моделью. В этом случае модель запускается прямо на устройстве пользователя. Это значит, что вы не отправляете данные через интернет в облако, а работаете напрямую с моделью, которая уже установлена в системе.

Мобильное приложение в этом случае взаимодействует с моделью через локальный интерфейс. Чаще всего это какой-то внутренний сервер или библиотека, которая подключается к приложению. Вы можете передавать запросы к модели прямо в файл или через локальный порт. Приложение формирует prompt, как и раньше, но теперь оно передаёт его не по сети, а напрямую в локальный движок модели. Ответ от модели тоже приходит мгновенно, без задержек от сети.

Для вас это даёт больше контроля. Вы можете заранее подготовить набор промтов, которые точно будут работать на этой конкретной версии модели. Вам не нужно беспокоиться о сетевых сбоях или ограничениях по скорости. Но при этом вы должны учитывать, что на мобильном устройстве мощности ограничены, и модель должна быть максимально лёгкой и оптимизированной. Вы также работаете совместно с разработчиками, чтобы интеграция модели прошла корректно, а интерфейс быстро реагировал на действия пользователя. Ваша задача — проектировать такие промты, которые хорошо себя ведут в условиях ограниченных ресурсов, и дают быстрые и понятные ответы прямо на устройстве. Это может работать в офлайн-режиме, ведь вся логика и взаимодействие с ИИ происходят внутри самого приложения, без подключения к внешним серверам.

7. Примеры использования

LLM в мобильных приложениях открывают множество новых возможностей, и вы, как prompt инженеры, играете ключевую роль в том, как эти возможности реализуются. Один из самых распространённых примеров — это чат-боты. Здесь модель выступает как собеседник, который отвечает на вопросы, помогает с заказами, подсказывает информацию. Вы проектируете промты так, чтобы бот вёл себя дружелюбно, точно отвечал на запросы и не путался в логике разговора. Важно заранее продумать, как бот будет реагировать на разные темы, ошибки и нестандартные ситуации.

Голосовые ассистенты работают по похожему принципу, но вместо текста используется речь. Пользователь говорит, вы с помощью

распознавания превращаете голос в текст, отправляете его как prompt в модель, получаете ответ и превращаете его обратно в речь. Здесь ваша задача — сделать так, чтобы модель понимала контекст и давала короткие, уместные ответы, потому что в голосовом интерфейсе важно не перегружать пользователя.

Генерация текста — это ещё одна сильная сторона LLM. Приложение может помочь пользователю написать отзыв, составить письмо, придумать заголовок или пост для соцсетей. Вы настраиваете промт таким образом, чтобы стиль текста подходил под ситуацию, был грамотным и звучал естественно. Вы также можете использовать подсказки, чтобы пользователь мог выбрать тональность или длину текста.

Персонализация и анализ — это кейс, в котором модель помогает адаптировать контент под конкретного пользователя. Например, приложение может анализировать, как человек пишет, какие слова он чаще использует, и подстраивать ответы под его стиль. Или давать советы на основе предыдущих действий. Вы создаёте промты, которые позволяют модели учитывать эти данные, сохранять тон общения, учитывать цели пользователя и делать взаимодействие более живым и точным. Всё это требует тонкой настройки и постоянного тестирования, чтобы ответы были максимально полезными и чувствовали личную направленность.

Telegram — это отличная платформа для запуска AI-ботов в мобильном приложении, особенно если вы хотите быстро протестировать идею или дать пользователю привычный способ общения. У Telegram есть открытое и понятное API, которое легко

подключается к любому серверу. Вы можете буквально за несколько часов создать полноценного бота, который будет принимать сообщения, отправлять их в LLM, получать ответ и возвращать его обратно пользователю. Всё работает через интернет, по обычным запросам, без сложных настроек.

Для пользователя Telegram уже привычен. Он знает, как устроен чат, как отправлять сообщения, как использовать кнопки. Вам не нужно объяснять, как работает интерфейс. Это снижает барьер входа и делает взаимодействие более естественным. Вы как prompt инженер можете сосредоточиться на логике диалога и качестве ответов, а не на построении интерфейса.

Кроме того, Telegram уже даёт вам всю инфраструктуру общения. У вас сразу есть история сообщений, форматирование текста, поддержка изображений, кнопки, команды. Вы можете использовать всё это, чтобы сделать общение с моделью более удобным. Например, вы можете задавать тональность ответа с помощью кнопок, предлагать быстрые команды, отображать варианты выбора. Вам не нужно строить всё с нуля. Telegram уже всё подготовил, и вы просто внедряете свои промты и логику поверх готовой среды. Это сильно ускоряет разработку и упрощает тестирование вашей идеи.

8. Роль prompts

Prompts — это основа поведения телеграм-бота, именно вы, как prompt инженеры, решаете, каким будет этот бот, как он будет говорить, что ему позволено, а что нет. Первый и самый важный prompt — это системное сообщение, которое задаёт контекст для модели. В

нём вы описываете, кто этот бот, какую роль он выполняет, как он должен себя вести. Например, вы можете написать, что это вежливый помощник, который объясняет сложные вещи простым языком, не шутит, не спорит и не выходит за рамки профессиональной тематики. Или наоборот — это может быть лёгкий, дружелюбный собеседник, который может поддержать разговор и даже пошутить.

Через prompts вы формируете личность бота. Модель каждый раз смотрит на этот контекст и строит свои ответы в заданном стиле. Вы задаёте, обращается ли бот на «вы» или на «ты», использует ли он смайлики, как он реагирует на грубость, отвечает ли на провокации. Это как сценарий, по которому бот играет свою роль. Без чёткого prompt-а модель будет отвечать случайно, без устойчивого поведения.

Также prompts определяют границы общения. Вы можете чётко указать, о чём бот говорит, а о чём — нет. Например, если бот должен помогать с медицинскими вопросами, вы можете ограничить его рамками общей справочной информации и запретить давать конкретные диагнозы. Если бот работает в юридической сфере, вы можете задать стиль делового общения и запрет на советы, которые выходят за рамки закона. Всё это прописывается в prompt, и чем чётче вы его составите, тем стабильнее будет поведение бота.

По сути, вы создаёте не просто ответы, а целый образ. Вы обучаете модель, как себя вести в рамках вашего приложения. Поэтому работа с prompt-ами — это не просто текст, а управляемая стратегия, с помощью которой вы формируете опыт пользователя и задаёте правила взаимодействия.

9. Архитектура Telegram бота

Когда вы создаёте Telegram-бота, который работает с LLM, у приложения появляется чёткая архитектура, в которой каждый компонент выполняет свою роль. В центре всего — Telegram Bot API. С его помощью ваш бот получает сообщения от пользователей и отправляет им ответы. Для работы с этим API удобно использовать библиотеку python-telegram-bot. Она упрощает обработку сообщений, кнопок, команд и позволяет быстро запускать бота.

Но сам бот — это не вся система. Здесь вы, как prompt инженер, заранее прописываете, как этот prompt должен выглядеть. Например, вы добавляете системное сообщение, стиль общения, ограничения по теме и вставляете туда текст от пользователя.

Дальше готовый prompt отправляется в LLM. Это может быть облачная модель, например, ChatGPT через REST API, или локальная модель, если вы используете что-то вроде llama.cpp. Вы отправляете запрос, получаете ответ от модели, и передаёте этот ответ обратно в Telegram через библиотеку python-telegram-bot. Пользователь получает ответ прямо в чат.

Таким образом, Telegram общается с LLM, а вы управляете всей логикой взаимодействия через prompts. Вы задаёте, как должна вести себя модель, как она должна отвечать, и как реагировать на разные типы сообщений. Архитектура остаётся простой, гибкой и позволяет быстро вносить изменения как в тексты, так и в поведение.

10. Заключение

Сегодня мы с вами рассмотрели важную и актуальную тему интеграции искусственного интеллекта в мобильные приложения. Давайте подведем итоги нашего обсуждения.

Мы изучили ключевые аспекты мобильной разработки. Вы узнали о преимуществах и особенностях создания как нативных приложений, так и решений на кроссплатформенных фреймворках. Каждый подход имеет свои сильные стороны и области применения.

Особое внимание было уделено архитектуре **REST API** и работе с локальными языковыми моделями. Эти знания позволят вам эффективно встраивать решения на основе ИИ в мобильных приложениях, обеспечивая высокую производительность и безопасность.

Telegram как платформа продемонстрировал вам возможности быстрого прототипирования и развертывания чат-ботов. Мы разобрали архитектуру таких ботов и роль промптов в их работе.

Помните, что успешная интеграция ИИ в мобильные приложения требует комплексного подхода:

- Понимание бизнес-задач и технологий разработки
- Навыки prompt engineering
- Знание принципов работы систем ИИ

Рекомендую продолжить практическое освоение полученных знаний, экспериментируя с различными промптами - это фундамент успешной работы в области разработки диалоговых систем.

Благодарю за внимание!