Ростелеком

3.3.2 Интеграция с telegram

Модуль 3. Интеграция приложений и prompt engineering

Оглавление

1. Введение

Сегодня мы с вами рассмотрим практическую составляющую интеграции искусственного интеллекта в Telegram-бота. Это направление становится все более востребованным, поскольку позволяет создавать умные автоматизированные системы для решения различных задач: от простых помощников до сложных аналитических инструментов.

Telegram, как одна из самых популярных мессенджеров в мире, предоставляет отличные возможности для разработки и внедрения AI-решений. Комбинация удобного API мессенджера и возможностей современных языковых моделей открывает широкие перспективы для создания инновационных продуктов и сервисов.

В ходе нашей лекции мы разберем весь процесс создания бота: от регистрации и настройки базового функционала до внедрения продвинутых техник работы с промптами. Вы узнаете, как правильно интегрировать OpenAI API, формировать эффективные запросы и обеспечивать безопасность вашего бота.

Особое внимание уделим различным стилям общения бота с пользователем: от четких инструкций до естественного диалога. Также рассмотрим продвинутую технику Chain-of-Thought, которая позволяет создавать более умные и эффективные AI-решения.

Понимание этих принципов и техник поможет вам создавать качественных ассистентов, которые будут полезны пользователям и востребованы на рынке. Полученные знания станут важным шагом в освоении профессии промпт-инженера и помогут вам реализовать собственные проекты.

2. Создание Telegram-бота через BotFather

Для создания Telegram-бота вам нужно найти в мессенджере специального бота под названием @BotFather. Это официальный бот платформы, который помогает создавать ботов и управлять ими. Введите @BotFather в поиске Telegram, откройте диалог с ним и нажмите кнопку Start для начала работы.

В диалоге с BotFather введите команду /newbot — это запустит процесс создания нового бота. BotFather попросит вас придумать имя для вашего бота. Помните, что имя должно заканчиваться на -bot и быть уникальным в системе Telegram. После того как вы выберете и подтвердите имя, BotFather выдаст вам токен доступа — это секретный ключ, который позволяет управлять вашим ботом. Храните его в надёжном месте, он понадобится для настройки и работы с ботом.

После получения токена ваш бот автоматически появится в списке контактов Telegram. Вы можете сразу начать с ним диалог, проверить его работу или перейти к настройке через BotFather, используя команду /mybots. Здесь вы сможете изменить описание бота, добавить аватарку или удалить его при необходимости.

Теперь ваш бот готов к дальнейшей настройке и интеграции с LLM-моделями. Токен, выданный BotFather, будет использоваться в коде для подключения вашего бота к Telegram API и обработки сообщений от пользователей.

3. Настройка обработчика сообщений

Давайте разберем, как настроить обработчик сообщений для вашего Telegram-бота. В коде мы используем библиотеку python-telegram-bot, которая помогает нам обрабатывать различные типы сообщений от пользователей.

Сначала мы импортируем необходимые компоненты: Application для создания основного приложения, CommandHandler для обработки команд, ContextTypes для работы с контекстом, MessageHandler для обработки обычных сообщений и filters для настройки фильтров сообщений.

Далее мы создаем основное приложение, передавая ему токен вашего бота из переменных окружения. Это как бы запускает ваш бот и делает его готовым к приему сообщений.

Затем мы настраиваем обработчики. Например, chat_handler отвечает за обработку текстовых сообщений, которые не являются командами. Фильтры TEXT & ~filters.COMMAND говорят, что мы хотим обрабатывать только обычные текстовые сообщения, исключая команды типа /start или /help.

После этого мы добавляем обработчики команд. CommandHandler(“start”, start) означает, что когда пользователь вводит команду /start, будет вызвана функция start, которую вы должны определить отдельно. Аналогично настраивается обработка команды /help.

И наконец, мы добавляем наш основной обработчик сообщений chat_handler, который будет реагировать на все обычные текстовые

сообщения от пользователей. Это позволяет вашему боту вести диалог и отвечать на сообщения в чате.

Таким образом, вы создаете систему, где бот может как выполнять команды, так и вести обычный диалог с пользователями, что особенно важно для работы с LLM-моделями.

4. Интеграция OpenAI API

Давайте разберем, как происходит интеграция с OpenAI API для работы с LLM-моделями. Сначала мы импортируем необходимые библиотеки: openai для работы с API и dotenv для работы с переменными окружения. Это позволяет нам безопасно хранить API-ключ в файле .env.

Далее мы создаем класс LLMService, который будет отвечать за взаимодействие с языковой моделью. В конструкторе класса мы инициализируем клиента OpenAI, передавая ему API-ключ из переменных окружения и URL сервера. Также здесь мы задаем системный промпт, который определяет поведение модели – в данном случае это роль оператора техподдержки, который должен отвечать вежливо.

Метод chat отвечает за отправку сообщений модели и получение ответа. При вызове этого метода мы формируем запрос к API, передавая в нем системный промпт и сообщение от пользователя. Также мы указываем параметры: модель для работы (в данном случае используется YandexGPT Lite), уровень случайности ответов (temperature) и максимальное количество токенов в ответе.

Когда модель обрабатывает запрос, она возвращает ответ, который мы извлекаем из полученного объекта и возвращаем обратно. В случае возникновения ошибки обработка оборачивается блоком try-except, что позволяет корректно обрабатывать возможные проблемы при работе с API.

Таким образом, этот код создает удобный интерфейс для взаимодействия с LLM-моделью через API, который можно легко интегрировать в Telegram-бота для обработки сообщений пользователей.

5. Подстановка prompt в запрос

Давайте разберем, как правильно работать с промптами и настройками в запросах к языковой модели. В примере вы видите создание запроса к API, где первым делом указывается модель для работы - в данном случае это YandexGPT Lite с определенным идентификатором.

Далее идет массив сообщений, где каждое сообщение представляет собой диалог. Первое сообщение с ролью system задает контекст и правила поведения модели - здесь указано, что модель должна вести себя как оператор техподдержки и отвечать вежливо. Второе сообщение с ролью user содержит сам вопрос или проблему от пользователя - в примере это жалоба на неработающий интернет.

Теперь о важных параметрах настройки. Температура (temperature) определяет креативность ответов модели. Значение 1.0 означает максимальную креативность, при этом ответы могут быть более

разнообразными, но менее предсказуемыми. Чем ниже значение, тем более последовательными и предсказуемыми будут ответы.

Max_tokens отвечает за максимальную длину ответа модели. В примере установлено 1024 токена, что примерно соответствует 700-800 символам текста. Это позволяет модели давать развернутые ответы, но не слишком длинные.

Управление контекстом происходит через массив сообщений - вы можете добавлять больше сообщений для сохранения истории диалога, что поможет модели лучше понимать контекст разговора. Например, если пользователь задаст дополнительный вопрос по той же проблеме, вы можете добавить это сообщение в массив, чтобы модель видела всю историю общения.

Таким образом, правильно настраивая эти параметры и формируя промпты, вы можете контролировать поведение модели и качество генерируемых ответов в вашем приложении.

6. Инструкционный стиль

Давайте вспомним, как работает система ролей в промптах и почему это так важно для получения качественных ответов. В данном примере мы видим два типа сообщений: системное сообщение (system) и сообщение пользователя (user).

Системное сообщение задает контекст и правила поведения модели. Фраза “Ты оператор техподдержки, отвечай вежливо” дает модели четкое понимание, в какой роли она должна отвечать и какой стиль общения использовать. Это как если бы вы надели костюм оператора техподдержки и получили инструкцию по общению с

клиентами – модель точно знает, как себя вести и какие формулировки использовать.

Сообщение пользователя содержит сам вопрос или проблему. В примере это жалоба на неработающий интернет. Важно, что модель уже “знает”, что она оператор техподдержки, поэтому будет отвечать соответствующим образом – профессионально и вежливо.

Такой формат общения помогает модели лучше понять задачу и сгенерировать более релевантный ответ. Когда вы четко определяете роль модели через системное сообщение, вы как бы направляете её мышление в нужное русло. Это особенно важно при интеграции с Telegram-ботом, где пользователи ожидают получить профессиональную помощь в определенном стиле.

Помните, что системное сообщение можно делать более детализированным, добавляя конкретные инструкции по стилю общения, формату ответов и даже ограничения по содержанию. Это поможет получить именно тот результат, который вы ожидаете от модели.

При этом важно сохранять баланс – системное сообщение должно быть достаточно конкретным, чтобы направить модель, но не слишком длинным, чтобы не перегружать контекст. Оптимально формулировать его кратко и по существу, как в данном примере.

7. Диалоговый стиль

Давайте разберем, как правильно организовать диалоговый стиль общения через API. В диалоговом формате важно сохранять

последовательность сообщений и правильно формировать структуру запроса.

Представьте, что вы создаете чат-бот для Telegram, который ведет диалог с пользователем. Каждый запрос к модели должен содержать историю общения, чтобы ответы были осмысленными и последовательными.

В этом примере мы видим структуру диалога: сначала идет системное сообщение, задающее тон общения, затем чередуются сообщения пользователя и ассистента. Каждое новое сообщение продолжает предыдущий контекст.

Важно сохранять последовательность ролей: после system всегда идет user, затем assistant, и так далее. Это помогает модели понимать, кто что сказал, и корректно генерировать ответы.

При работе с Telegram-ботом вы должны сохранять историю диалога в памяти бота и добавлять новые сообщения в массив messages перед каждым запросом к модели. Это позволит модели видеть контекст разговора и давать осмысленные ответы, связанные с предыдущими сообщениями.

Не забывайте обновлять массив messages после каждого ответа модели, добавляя новое сообщение ассистента, чтобы следующий запрос учитывал весь предыдущий диалог. Это ключевой момент для создания естественного и связного общения с пользователем.

Такой подход поможет вам создать бота, который будет вести осмысленный диалог и не потеряет нить разговора при общении с пользователями.

8. Chain-of-Thought

Давайте разберем, как правильно использовать метод Chain-of-Thought при работе с API. Этот подход помогает модели лучше понимать задачу и более последовательно приходить к ответу.

В отличие от простого запроса, где мы просто получаем ответ, Chain-of-Thought заставляет модель подробно расписывать свои рассуждения. Это особенно полезно при решении сложных задач или, когда нужно объяснить логику принятия решения.

В нашем примере мы добавили важное требование в системное сообщение - подробно объяснять свои действия. Теперь модель не просто даст ответ, а сначала опишет процесс диагностики проблемы.

При работе с Telegram-ботом это особенно полезно, когда нужно не просто решить проблему пользователя, но и научить его самостоятельно справляться с подобными ситуациями в будущем.

Вы можете дополнительно уточнить формат рассуждений, например: “Сначала проведи базовую диагностику, затем перечисли возможные причины, после этого предложи пошаговое решение”

Такой подход помогает получить более структурированный и понятный ответ. Модель будет следовать заданному формату рассуждений, что делает процесс решения задачи более прозрачным для пользователя.

Важно помнить, что при использовании Chain-of-Thought нужно давать модели достаточно контекста и четко формулировать требования к формату рассуждений. Это поможет получить именно тот результат, который вы ожидаете.

При интеграции с Telegram учитывайте, что подробные рассуждения могут занимать больше токенов, поэтому следите за лимитами и при необходимости разбивайте ответ на несколько сообщений.

Такой подход особенно эффективен при работе со сложными техническими задачами, где важно не только решить проблему, но и объяснить пользователю, как он может сделать это самостоятельно в будущем.

9. Основы безопасности

Давайте поговорим о важных аспектах безопасности при работе с Telegram-ботом и LLM-моделями. Первое, на что стоит обратить внимание - это фильтрация ввода. Когда пользователи отправляют сообщения вашему боту, они могут содержать потенциально опасный или нежелательный контент. Поэтому важно проверять и очищать все входящие сообщения перед тем, как передавать их дальше.

Фильтрация может включать проверку на наличие вредоносного кода, удаление специальных символов, ограничение длины сообщений и проверку на соответствие правилам использования бота. Например, можно запретить отправку сообщений, содержащих нецензурную лексику или подозрительные ссылки.

Второй важный аспект - это rate limiting, или ограничение частоты запросов. Это необходимо для защиты как вашего бота, так и LLM-модели от перегрузки. Представьте ситуацию: если один пользователь начнет отправлять сотни сообщений в минуту, это может привести к сбоям в работе или даже к блокировке вашего бота.

Для реализации rate limiting вы можете отслеживать количество запросов от каждого пользователя за определенный период времени и ограничивать их, если они превышают допустимую норму. Например, можно разрешить не более 5 запросов в минуту от одного пользователя.

Также важно учитывать лимиты API LLM-модели и не отправлять слишком много запросов одновременно. Это поможет избежать ошибок и перебоев в работе. Рекомендуется добавить паузу между запросами или использовать очередь для обработки сообщений.

Не забывайте о защите конфиденциальных данных - храните API-ключи и токены в защищенном месте, используйте переменные окружения вместо того, чтобы писать их прямо в код. Это поможет избежать утечки важной информации.

Таким образом, правильная реализация фильтрации ввода и rate limiting сделает ваш бот более надежным и защищенным от возможных атак или случайного перегруза системой.

10. Заключение

Сегодня мы с вами прошли полный путь создания Telegram-бота с интеграцией искусственного интеллекта. Давайте подведем итоги нашего изучения.

Мы освоили ключевые этапы разработки:

- Создание и настройка бота через BotFather
- Настройка системы обработки сообщений
- Интеграция с OpenAI API
- Работа с промптами различных типов

- Обеспечение безопасности бота

Особое внимание мы уделили различным стилям общения бота с пользователем:

- Инструкционный стиль для четких указаний
- Диалоговый стиль для естественного общения
- Продвинутая техника Chain-of-Thought для более глубокого анализа

Практические навыки, полученные сегодня, позволят вам разрабатывать собственных Telegram-ботов или интегрировать ИИ в уже существующие проекты.

Рекомендую продолжить практику, экспериментируя с различными стилями общения и техниками работы с промптами. Особое внимание уделите отработке навыков составления эффективных запросов и оптимизации взаимодействия с языковыми моделями.

Благодарю за внимание!