Глава 6. Добавляем пользовательский интерфейс RAG с помощью Gradio

Это продолжение перевода книги Кит Борн. Раскрытие потенциала данных с помощью генеративного ИИ и технологии RAG. Глава 6 открывает вторую часть книги – **Компоненты RAG**. В этой части вы узнаете о ключевых компонентах системы RAG и о том, как их реализовать с помощью LangChain. Вы изучите, как создавать интерактивный пользовательский интерфейс в Gradio, вектора и векторные хранилища, методы оценки RAG и визуализацию. Вы научитесь использовать компоненты LangChain: загрузчики документов, разделители текста, парсеры вывода.

В этой главе мы предоставим практическое руководство по созданию интерактивного приложения на основе RAG с использованием Gradio в качестве пользовательского интерфейса. Мы настроим среду Gradio, интегрируем модель RAG, создадим web-интерфейс, и разместим приложения в Интернете. Вы узнаете, как быстро создавать прототипы и развертывать приложения на базе RAG.

Предыдущая глава Содержание Следующая глава

Существуют целые книги, написанные о том, как создать интерфейс, но мы предложим вам простой способ, используя Gradio. Это упрощает демонстрацию моделей. Мы раскроем следующие темы:

- Почему Gradio?
- Преимущества использования Gradio
- Ограничения использования Gradio
- Лаборатория кода 6.1 – Добавление интерфейса Gradio

Код для этой главы размещен в репозитории GitHub.

Почему Gradio?

До этого момента мы фокусировались на темах, которые обычно относятся к миру науки о данных. Машинное обучение, обработка естественного языка (NLP), генеративный искусственный интеллект (GenAI), большие языковые модели (LLM) и RAG — это технологии, которые требуют значительного опыта и отнимают много времени, так что мы можем и не быть специалистами в работа с веб-технологиями. Веб-разработка сама по себе является высокотехнической областью и требует значительного опыта и знаний для успешной реализации.

Тем не менее, в случае с RAG может быть очень полезно иметь пользовательский интерфейс, особенно если вы хотите протестировать его или продемонстрировать потенциальным пользователям. Как мы можем это обеспечить, если у нас нет времени на изучение веб-разработки? Это основная причина, по которой многие специалисты по обработке и анализу данных, включая меня, используют Gradio. Это позволяет быстро настроить и запустить пользовательский интерфейс с базовыми функциями аутентификации.

Мы ограничим обсуждение Gradio компонентами, которые помогут разместить приложение RAG в Интернете и поделиться им. Тем не менее, мы рекомендуем продолжить изучение Gradio, чтобы понять, есть ли что-то еще, что он может сделать для ваших приложений!

Преимущества использования Gradio

Во-первых, Gradio очень прост для тех, кто не является веб-разработчиком. Во-вторых, основная библиотека Gradio имеет открытый исходный код. Далее, Gradio хорошо интегрируется с широко используемыми фреймворками машинного обучения: TensorFlow, PyTorch и Keras. Gradio предлагает платформу, где разработчики могут развертывать интерфейсы своих моделей и управлять доступом. Gradio включает функции, которые упрощают совместную работу команд и сбор отзывов.

Gradio хорошо интегрируется с Hugging Face. Основанная бывшими сотрудниками OpenAI, Hugging Face имеет множество ресурсов, предназначенных для поддержки сообщества генеративного ИИ, таких как обмен моделями и хостинг наборов данных. Одним из ресурсов является возможность установить постоянную ссылку на вашу демонстрацию Gradio в Интернете с помощью Hugging Face Spaces. Hugging Face Spaces предоставляет инфраструктуру для постоянного размещения вашей модели машинного обучения бесплатно! Загляните на веб-сайт Hugging Face, чтобы узнать больше.

Ограничения на использование Gradio

Вероятно, самое важное, что следует помнить о Gradio, это то, что он не обеспечивает адекватной поддержки для создания приложения производственного уровня, которое будет взаимодействовать

с сотнями, тысячами или даже миллионами пользователей. В этом случае вы, вероятно, захотите нанять кого-то с опытом создания фронтендов для крупномасштабных приложений. Но для того, чтобы протестировать ваше приложение с пользователями с базовой интерактивностью, Gradio проделывает фантастическую работу.

Еще одно ограничение – отсутствие гибкости в том, что вы можете создать. Если вы или ваши пользователи начнете требовать более сложных функций пользовательского интерфейса, Gradio будет гораздо более ограничивающим, чем полноценная среда веб-разработки.

Лаборатория кода 6.1 – Добавление интерфейса Gradio

Продолжите код с того места, на котором мы остановились в главе 5, за исключением последнего набора строк, представляющих атаку с помощью prompt probe. Начнем с установки нового пакета Gradio. Мы также собираемся удалить uvloop из-за конфликта с другими пакетами:

```
%pip install gradio
%pip uninstall uvloop -y
```

Далее мы добавим несколько пакетов в список импорта:

```
import asyncio
import nest_asyncio
asyncio.set_event_loop_policy(asyncio.DefaultEventLoopPolicy())
nest_asyncio.apply()
import gradio as gr
```

Эти строки импортируют библиотеки asyncio и nest_asyncio и настраивают политику циклов событий. Asyncio — библиотека для написания параллельного кода с использованием сопрограмм и циклов событий. nest_asyncio — библиотека, которая позволяет создавать вложенные циклы событий в записной книжке Jupyter. Политика asyncio.set_event_loop_policy(asyncio.DefaultEventLoopPolicy()) устанавливает политику цикла событий в соответствии с политикой по умолчанию. nest_asyncio.apply() применяет необходимые патчи для включения вложенных циклов событий. Затем мы импортируем пакет gradio и присваиваем ему псевдоним gr.

После добавления импортов нам нужно добавить следующий код в последнюю ячейку для настройки интерфейса Gradio:

```
def process_question(question):
  result = rag_chain_with_source.invoke(question)
  relevance_score = result['answer']['relevance_score']

  final_answer = result['answer']['final_answer']
  sources = [doc.metadata['source'] for doc in result['context']]
  source_list = ", ".join(sources)
  return relevance_score, final_answer, source_list
```

Функция process_question — это функция, которая вызывается при нажатии кнопки «Отправить». Вы определите этот вызов в коде gr.Interface. Функция process_question принимает вопрос, который вы отправляет пользователь и обрабатывает его с помощью конвейера RAG. Он вызывает объект rag_chain_with_source с заданным вопросом и извлекает из результата оценку релевантности, окончательный ответ и источники. Затем функция объединяет источники в строку, разделенную запятыми, и возвращает оценку релевантности, окончательный ответ и список источников.

Далее мы настроим экземпляр интерфейса Gradio:

```
demo = gr.Interface(
  fn=process_question,
  inputs=gr.Textbox(label="Enter your question",
    value="What are the Advantages of using RAG?"),
  outputs=[
    gr.Textbox(label="Relevance Score"),
    gr.Textbox(label="Final Answer"),
    gr.Textbox(label="Sources")
```

```
    ],
    title="RAG Question Answering",
    description=" Enter a question about RAG and get an answer, a
       relevancy score, and sources."
)
```

demo = gr.Interface(…) – это место, где происходит магия Gradio. Он создает интерфейс Gradio с помощью команды gr.Interface. fn определяет функцию, которая должна быть вызвана, когда пользователь взаимодействует с интерфейсом. fn вызывает process_question, запуская конвейер RAG. Параметр inputs определяет входной компонент интерфейса, которым является gr.Textbox для ввода вопроса. Параметр outputs определяет выходные компоненты интерфейса, которые используют три gr.Textbox для отображения оценки релевантности, ответа и источников. Параметры title и description задают заголовок и описание интерфейса.

Осталось запустить интерфейс:

```
demo.launch(share=True, debug=True)
```

Параметр share=True включает функцию общего доступа Gradio, генерируя общедоступный URL, которым вы можете поделиться с другими для доступа к интерфейсу. Gradio использует службу туннелирования для обеспечения этой функциональности, позволяя любому, у кого есть URL-адрес, взаимодействовать с интерфейсом без необходимости запускать код локально. Параметр debug=True включает режим отладки, предоставляя дополнительную информацию и инструменты для отладки и разработки. В режиме отладки Gradio отображает сообщения в консоли браузера, если во время выполнения функции process_question возникают какие-либо ошибки .

Я считаю demo.launch(share=True, debug=True) особой строкой кода по сравнению со всем остальным кодом, который вы написали в этой книге. Это потому, что он делает то, чего вы раньше не видели; он запрашивает Gradio для запуска локального веб-сервера для размещения интерфейса, определенного в gr.Interface(…). Когда вы запускаете ячейку, вы заметите, что она продолжает работать бесконечно, пока вы ее не остановите. Вы также заметите, что вы не можете запустить другие ячейки, не остановив код.

Есть еще один необязательный параметр, который нужно добавить в demo.launch:

```
demo.launch(share=True, debug=True, auth=("admin", "pass1234"))
```

auth добавит простой уровень аутентификации в случае, если вы публикуете свое приложение в открытом доступе. Он генерирует дополнительный интерфейс, для которого требуется имя пользователя (admin) и пароль (pass1234). Измените admin/pass1234 на что угодно, но обязательно измените! Предоставьте доступ к этим учетным данным только тем пользователям, которым вы хотите предоставить доступ к приложению RAG. Имейте в виду, что это не очень безопасно, но это служит по крайней мере базовой цели для ограничения доступа пользователей.

Теперь у вас есть активный веб-сервер, который принимает входные данные, обрабатывает их, реагирует и возвращает новые элементы интерфейса на основе кода, который вы написали для интерфейса Gradio. Это то, что раньше требовало значительного опыта в веб-разработке, но теперь вы можете настроить и запустить его за считанные минуты! Это позволит вам сосредоточиться на написании кода для приложения RAG!

После ввода кода Gradio интерфейс становится интерактивным, позволяя вводить запросы. Когда пользователь отправляет вопрос, функция process_question вызывается с вопросом пользователя в качестве входных данных. Функция вызывает конвейер RAG rag_chain_with_source с вопросом и извлекает оценку релевантности, окончательный ответ и источники из результата. Затем он возвращает оценку релевантности, окончательный ответ и список источников. Gradio обновляет выходные текстовые поля возвращенными значениями, отображая пользователю оценку релевантности, окончательный ответ и источники.

Интерфейс остается активным и отзывчивым до тех пор, пока не завершится выполнение ячейки или пока не будет вызван gr.close_ all() для закрытия всех активных интерфейсов Gradio.

Когда вы запустите ячейку в Jupyter Notebook с кодом demo.launch(…), появится окно:

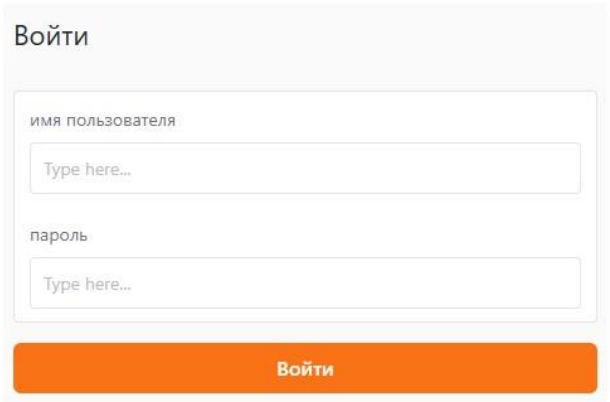

Рис. 6.1. Аутентификация

Над этим интерфейсом в Jupyter Notebook вы увидите текст, похожий на этот:

```
demo.launch(share=True, debug=True, auth=("###", "###"))

Rerunning server... use `close()` to stop if you need to change `launch()` parameters.
----
Colab notebook detected. This cell will run indefinitely so that you can see errors and
logs. To turn off, set debug=False in launch().
* Running on public URL: https://45128d2e9f2c2ee905.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades,
run `gradio deploy` from the terminal in the working directory to deploy
to Hugging Face Spaces (https://huggingface.co/spaces)
```

Рис. 6.2. Сообщение в Jupyter Notebook при запуске demo.launch(…)

Нажав на первую ссылку, вы попадете в интерфейс в окне браузера! Он будет выглядеть так же, как на рисунке 6.3, но займет все окно браузера. Оставаясь в блокноте вы можете ввести логи и пароль. Откроется интерфейс Gradio:

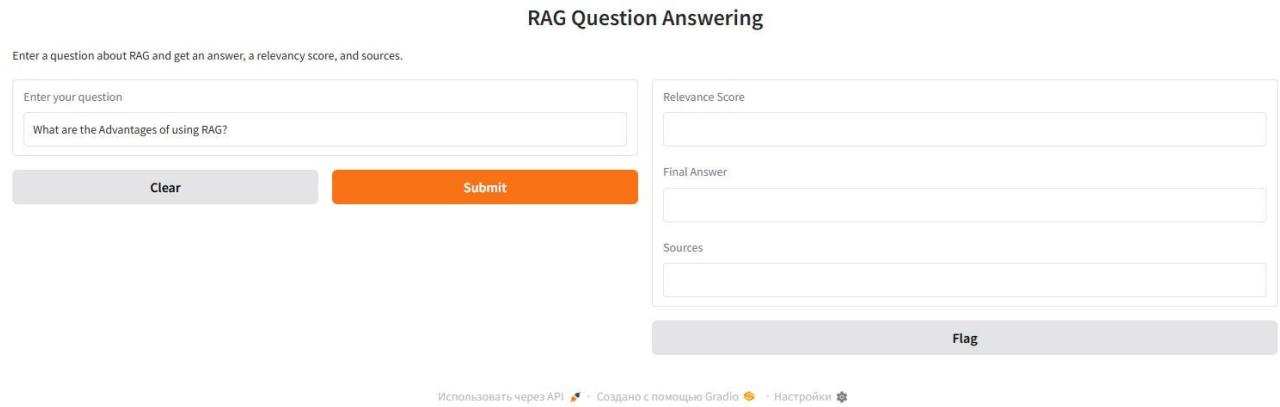

Рис. 6.3. Интерфейс Gradio

Нажатие *Отправить* запускает процесс RAG, передавая то, что вы вводите в качестве вопроса, в цепочку LangChain с result = rag_chain_with_source.invoke(question) и возвращая ответ после ожидания в несколько секунд:

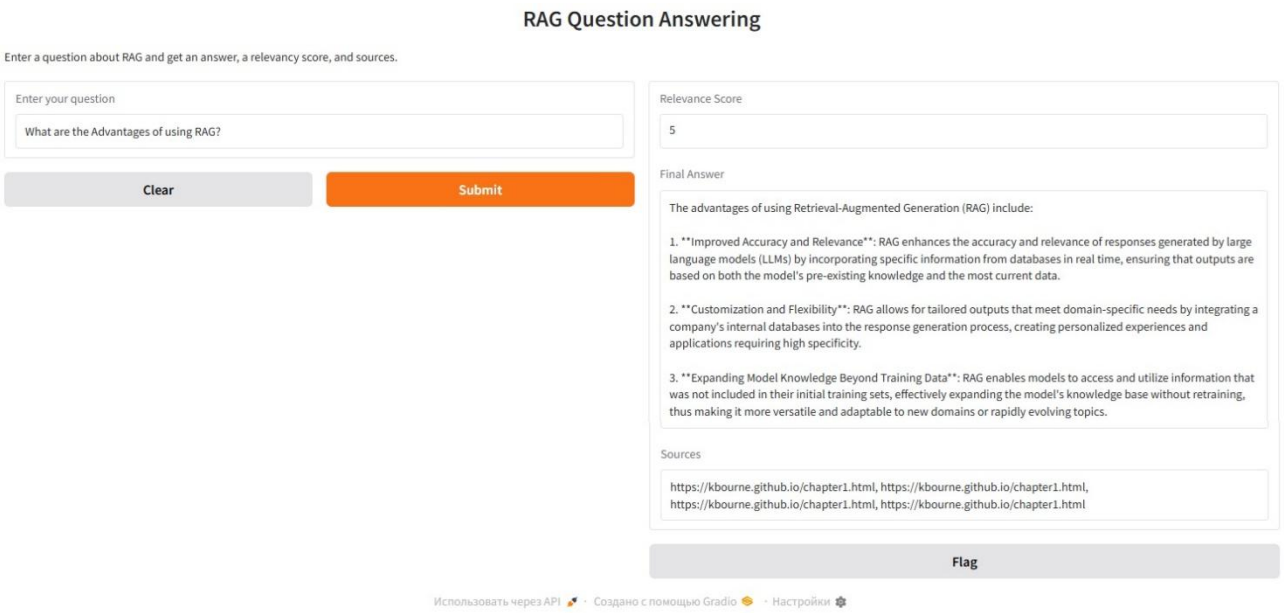

Рис. 6.4. Интерфейс Gradio с откликом

Мы предварительно заполнили поле вопроса: Каковы преимущества использования RAG? Вы можете изменить этот вопрос и задать что-то другое. Если это не имеет отношения к содержимому базы данных, LLM ответит «Я не знаю». Мы рекомендуем вам попробовать актуальные, и неактуальные вопросы! Посмотрите, сможете ли вы найти вопросы, которые позволят вам улучшить навыки отладки.

Ответ из ChatGPT 4 уже был отформатирован таким образом, что отображается с разметкой. Gradio автоматически будет использовать переносы строк из этой разметки и выводить текст, разбивая абзацы. Источники представляют собой список из четырех источников, указывающий на то, что в ретривере было четыре источника. Этот код мы настроили в главе 3, когда добавили возможность переноса источников полученных результатов в метаданных, чтобы они были доступны для отображения в пользовательском интерфейсе. Возможно, вы заметили, что все четыре источника одинаковы. Это результат того, что это небольшой пример, в котором мы работаем с один источником.

Это было всего лишь знакомство с Gradio. Мы рекомендуем вам зайти на веб-сайт Gradio и ознакомиться с их кратким руководством и документацией, чтобы узнать о других важных функциях, предоставляемых их платформой.