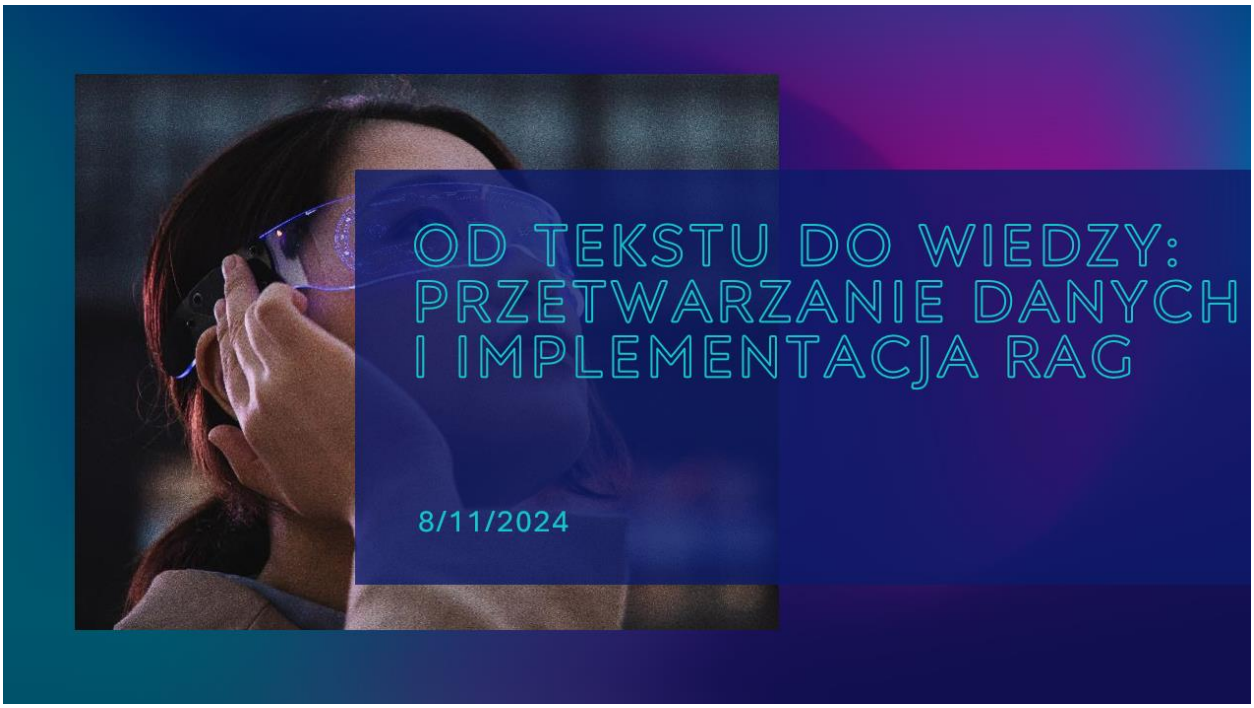


Cel ćwiczenia

Celem tego ćwiczenia jest poznanie techniki Retrieval Augmented Generation (RAG) służącej do zwiększania skuteczności dużych modeli językowych (ang. Large Language Model - LLM). Naszym zadaniem będzie wzbogacenie istniejącego modelu LLM o nowe dane pochodzące ze sprawozdania finansowego firmy Nike za 2023 rok.

Dlaczego RAG?

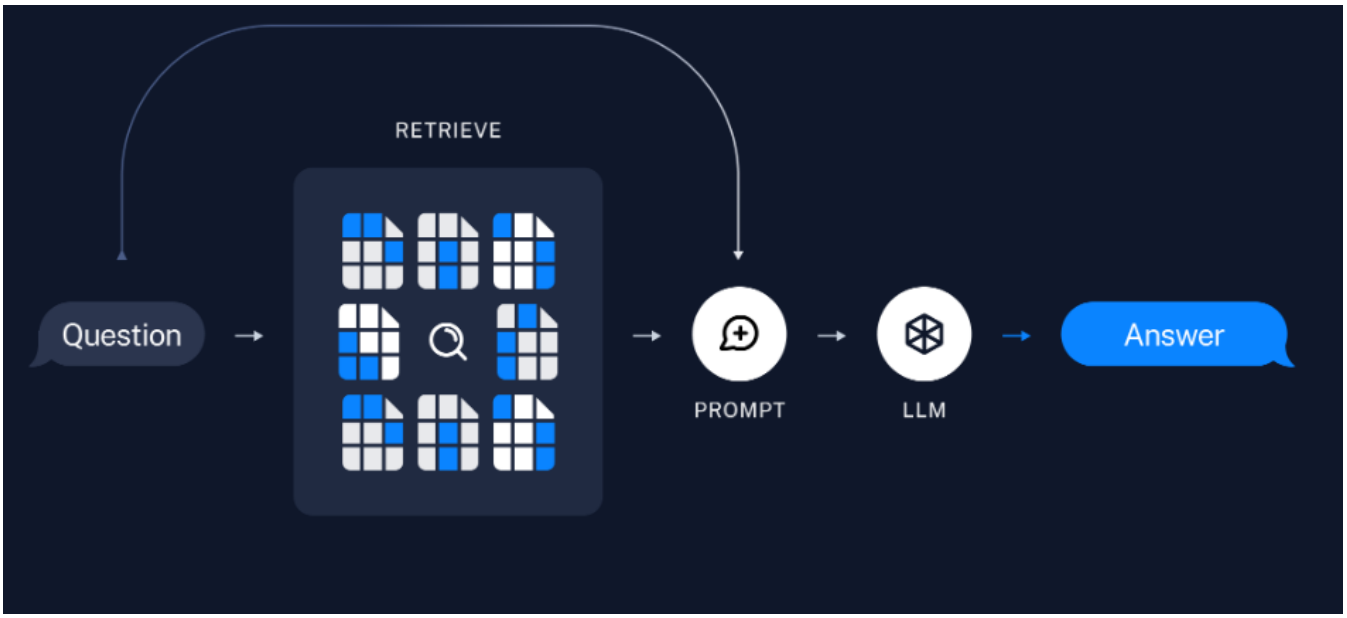
Aby zwiększyć skuteczność modeli, w przypadku dotychczasowych modeli LLM, dostrajanie (ang. fine-tuning) było zawsze najlepszym rozwiązaniem. Dostrajanie wiąże się jednak z dodatkowym doszkalaniem z wykorzystaniem nowych danych, co może być procesem czasochłonnym i kosztownym.

Retrieval Augmented Generation (RAG) jest metodą wzbogacającą prompty (zapytania do modelu) poprzez dołączenie kontekstu zawierającego odpowiednie informacje z zewnętrznych źródeł danych. Ta technika zwiększa możliwości dużych modeli językowych w pracy z niestandardowymi danymi, umożliwia korzystanie ze stale aktualizowanych danych, co byłoby praktycznie nieosiągalne gdybyśmy chcieli uzywać fine-tuning.

2 Fazy podejścia RAG:

- Faza wyszukiwania – w fazie wyszukiwana po otrzymaniu zapytania wejściowego, modele korzystają z korpusu tekstowego lub bazy danych, aby znaleźć informacje lub dokumenty, które mogą zawierać potrzebne odpowiedzi.

- Faza generowania – w fazie generowania, system generuje odpowiedzi na podstawie pobranego tekstu. Na tym etapie, model łączy znalezione informacje z podstawową wiedzą uzyskaną z danych, na których był wytrenowany.



Rysunek 1 Jak działa RAG (źrodło: https://python.langchain.com/docs/tutorials/pdf_qa/)

Zalety:

- Dynamiczna baza wiedzy, możliwość reagowania na najnowsze informacje dotyczące specyficznej domeny

Wyzwania:

- Potencjalne wydłużenie czasu reakcji

Zadanie

Napisz aplikację chatbot, która będzie w stanie odpowiadać na szczegółowe pytania dotyczące sprawozdania finansowego firmy Nike.

Przygotowanie:

1. Zainstaluj Pythona w wersji 3.12: https://www.python.org/ftp/python/3.12.7/python-3.12.7-amd64.exe

2. Zainstaluj środowisko IDE, np. Pycharm lub VSCode (zajęcia będą prowadzone przy użyciu Pycharm): https://www.jetbrains.com/pycharm/download/download-thanks.html?platform=windows&code=PCC

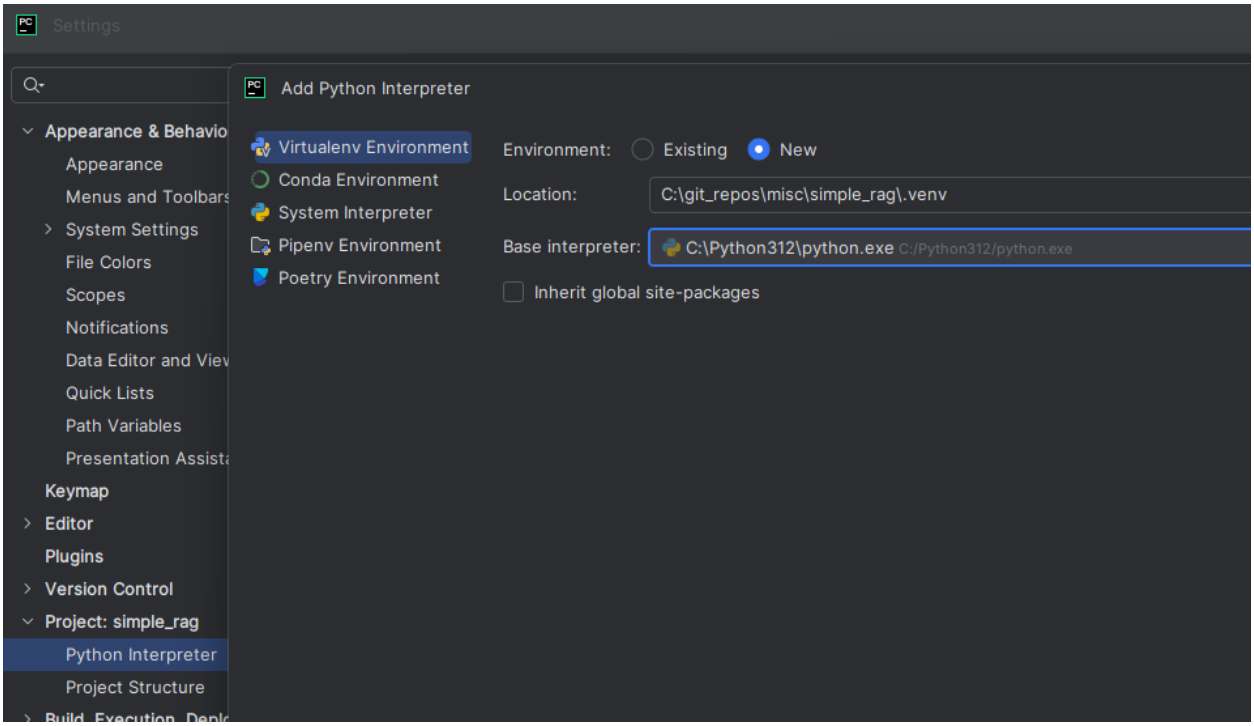
3. Dołącz do serwera discord: https://discord.gg/VpCndEc5, aby móc skopiować API key podczas zajęć

Instrukcja

1. Sklonuj repozytorium: https://github.com/hcymerys/ai_chat
2. Przygotuj venv i zainstaluj niezbędne biblioteki z pliku requirements.txt:
    a. Instrukcje opisujące jak stworzyć venv:

        - Instrukcja dla Pycharma

        - Instrukcja dla VS Code



Rysunek 2 Tworzenie venv w Pycharmie

b. Upewnij się, że venv jest aktywny (.venv widoczny oknie Terminala), użyj komendy:

```
(.venv) PS C:\git_repos\misc\ai_chat> .\.venv\Scripts\activate
```

3. Połącz się z modelem OpenAI przy użyciu przekazanego klucza API:
   a. Dodaj plik *.env* o treści: *OPEN_API_KEY=<api_key>* w katalogu głównym projektu
   b. WAŻNE: w ramach ćwiczenia używajmy modelu **"*gpt-4o-mini*"**
   c. Ten sam klucz API posłuży nam także do użycia modelu embeddingowego: „text-embedding-3-large” (OpenAIEmbeddings)
4. Uruchom przykładowy skrypt: *python simple_chat.py* aby przetestować połączenie z modelem (Sprawdź czy skrypt odpowiada na pytanie "What was Nike's revenue in 2023?")

(Dalsze kroki w oparciu o instrukcję: https://python.langchain.com/docs/tutorials/pdf_qa/)

5. Załaduj plik PDF sprawozdania finansowego firmy Nike

   - Dostępne loadery: Document loaders | 🦜🔗 LangChain)

   - Rekomendowane podejście: PyPDFLoader | 🦜🔗 LangChain

6. Podziel dokument na mniejsze części:

- Przykład użycia: How to recursively split text by characters | 🦜🔗 LangChain

Checkpoint: Sprawdź liczbę stron i zawartość strony pierwszej przy użyciu Loadera

7. Stwórz wektorową bazę danych w oparciu o dane pochodzące z wczytanego pliku PDF.

   - Dostępne bazy danych: Vectorstores | 🦜🔗 LangChain)

   - Rekomendowane podejście: InMemoryVectorStore — 🦜🔗 LangChain documentation

   - *ChromaDB: Chroma | 🦜🔗 LangChain (dotyczy zadania dodatkowego 12d.)

Checkpoint: Sprawdź, czy baza zwraca chunks związane z zapytaniem (czy posiadają zawartość podobną do treści zapytania)

8. Stwórz Retrieval w oparciu o bazę danych

   - Przykład użycia: How to use a vectorstore as a retriever | 🦜🔗 LangChain

9. Stwórz Retrieval chain w oparciu o bazę danych i systemowy prompt

   - Omówienie: Retrievers | 🦜🔗 LangChain

   - Przykład użycia: Build a Retrieval Augmented Generation (RAG) App | 🦜🔗 LangChain

Checkpoint: Sprawdź, czy chain odpowiada na pytanie z użyciem bazy danych

10. Przygotuj aplikację przyjmującą input z konsoli i zwracającą odpowiedzi na stdout (ekran) w pętli.
11. Aplikacja powinna być w stanie odpowiadać na następujące pytania:
    - "What was Nike's revenue in 2023?"
    - "Who is Craig Williams?"
    - "Tell me about gross margin of Nike in 2023"
    - "Tell me revenues in footwear, apparell and equipment for Nike in 2023, 2022 and 2021"

12. Dodatkowe zadania:
    a. Do odpowiedzi aplikacji dodaj kontekst informujący o źródle danych (strona, fragment tekstu z PDF)
       - How to get your RAG application to return sources | 🦜🔗 LangChain
    b. Sprawdź jak będą wyglądały odpowiedzi dla parametrów: *chunk_size=100, chunk_overlap=0*

c. Sprawdź jak wyglądają wektory w bazie
d. Zmień bazę wektorową z *InMemoryVectorStore* na *ChromaDB* (załadowanie raz i czytanie z bazy)