

Politechnika Krakowska
Wydział Informatyki
i Telekomunikacji

Politechnika Krakowska
Wydział Informatyki i Telekomunikacji
Retrieval Augmented Generation (RAG)

Miłosz Hańczyk

18 listopada 2023

1	Abstrakt
2	Wstęp
3	Metodologia
3.1	Zbiór danych
3.2	Łańcuch przetwarzania
4	Wyniki i Dyskusja
4.1	Metody weryfikacji wyników
4.2	Aplikacja webowa
5	Podsumowanie
6	Bibliografia

1 Abstrakt

Raport "Retrieval Augmented Generation" (RAG) skupia się na najnowszych badaniach i praktykach dotyczących integracji systemów wyszukiwania i generacji tekstów w ramach jednego modelu (aplikacji). RAG jest konceptem, który zakłada wykorzystanie dużych modeli językowych do przetwarzania dokumentów. Użytkownik może zadać pytanie odnośnie załączonego przez niego dokumentu, a następnie otrzymać odpowiedź ze wskazaniem fragmentu dokumentu, z którego model językowy skorzystał generując ją. W raporcie omawiamy kluczowe koncepcje, technologie i studia przypadków związane z RAG oraz jego implementację z użyciem lokalnego, bądź zdalnego modelu językowego.

2 Wstęp

Wobec rosnącej popularności wykorzystania sztucznej inteligencji do tworzenia dialogów i odpowiedzi na pytania, efektywne tworzenie odpowiedzi jest kluczowe. RAG to nowy model tworzenia systemów wyciągania wiedzy (IR, Information Retrieval) z dziedziny systemów językowych, wykorzystujący jednocześnie modele generatywne i modele embedding'owe do wytworzenia bardziej trafnych odpowiedzi na zadane pytania. Dzięki przeformułowaniu pierwotnego zastosowania dużych modeli językowych (LLM) oraz wykorzystaniu pomocniczego modelu typu embedding, RAG stanowi niejaki dopełnienie do zasadniczej wady korpusów językowych, jaką jest ograniczenie ich wiedzy do zbioru uczącego. Problem ten jest omijany poprzez potraktowanie modelu embedding'owego, trenowanego zgodnie z celem minimalizacji odległości w przestrzeni między podobnymi zdaniami, jako wstępnego filtru dokumentów, a następnie użycie modelu językowego, jako generatora odpowiedzi zrozumiałych dla człowieka, zawierających odpowiedź na zadane pytanie bazujące na wartościach zwróconych w poprzednim kroku.

3 Metodologia

3.1 Zbiór danych

RAG wykorzystuje zbiór danych w postaci tekstu ciągłego, zapisanych jako dokumenty na konkretny temat, takie jak np. raport ze sprzedaży, raport finansowy, regulaminy, opisy organizacji, instrukcje itp. Każdy z zadanych dokumentów jest następnie dzielony na fragmenty (ang. chunks) zgodnie z przyjętą polityką dzielenia, która zazwyczaj daje najlepsze rezultaty, jeśli jest dostosowana pod konkretne zastosowanie.

```
def process_docs(docs):
    text_splitter = RecursiveCharacterTextSplitter(
        chunk_size=500,
        chunk_overlap=100,
    )
    texts = text_splitter.split_documents(docs)
    return texts
```

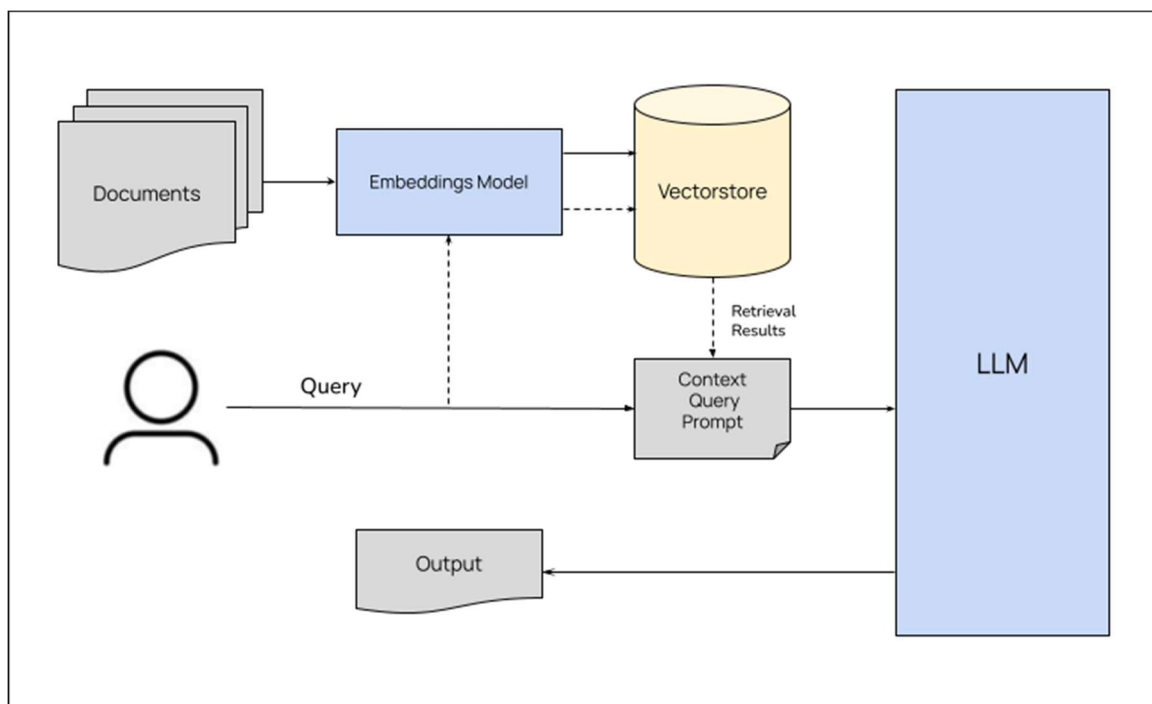
Rys. 1 Chunkowanie używające metody tworzenia fragmentów z paragrafów/akapitów o maks. długości 500 znaków

W celach demonstracji użyto zbioru dokumentów ze strony Politechniki Krakowskiej dotyczącej informacji o Szkole Doktorskiej. [Link](#)

W celu podzielenia dokumentów dot. Szkoły Doktorskiej na Politechnice Krakowskiej użyto metody rekurencyjnego dzielenia, która stara się z każdego akapitu tworzyć chunk.

3.2 Łańcuch przetwarzania

System RAG jest składa się z 2 komponentów: modelu embedding’owego oraz modelu językowego. W podanej kolejności są one w stanie wyciągnąć informację z dokumentów (retrieval) oraz wygenerować odpowiedź na zadane pytanie (generation). Sfragmentaryzowane dane są przekazywane do modelu embedding’ującego w celu przetworzenia każdego fragmentu na wektor liczb w pewnej przestrzeni oraz zapisania każdego z nich w wektorowej bazie danych. Przy podaniu zapytania do systemu RAG następuje przetworzenie go do postaci wektora liczb, a następnie wyszukanie listy k-podobnych do niego fragmentów w oparciu o zadaną metrykę podobieństwa, gdzie k jest parametrem systemu. Znaleziona lista fragmentów jest następnie podawana na wejście modelu językowego jako informacja kontekstowa wraz z zapytaniem użytkownika i poleceniami dla modelu językowego, które są kolejnym parametrem systemu. Cały prompt podawany jest jako wejście do modelu językowego, z którego następnie otrzymywana jest odpowiedź



Rys. 2 Architektura systemu RAG

```

template = """
You are assistant for Cracow University Of Technology related inquiries.
Answer the question based only on the following context and don't try
to make up an answer. If you don't know, just say you don't know.
Generate the answer for the question using only the sources below.

{context}

Question: {question}
"""
  
```

Rys. 3 Przykładowy szablon prompta z poleceniami dla modelu językowego

```
def init_chain(llm_provider, docs_path):
    print("Initializing chain...")
    llm, embeddings = load_llm(llm_provider)
    retriever = create_retriever(docs_path, embeddings)

    prompt = PromptTemplate.from_template(template)
    chain = (
        {"context": retriever | format_docs, "question": RunnablePassthrough()}
        | prompt
        | llm
        | StrOutputParser()
    )

    return chain
```

Rys. 4 Implementacja łańcucha przetwarzania z użyciem biblioteki Langchain

4 Wyniki i Dyskusja

4.1 Metody weryfikacji wyników

Weryfikacja systemów RAG jest wciąż tematem wymagającym badań nad rzetelnością wyników. Obecnie wykorzystywane są metody używające innych modeli językowych, które przyjmują query, odpowiedź systemu RAG oraz źródła, z których korzystał do wydania odpowiedzi TAK/NIE, a następnie obliczenia średniej skuteczności ze wszystkich promptów. Dodatkowo taki zbiór walidacyjny można zbudować poprzez komponent generujący query do fragmentów tekstów, który jest oparty o inny model językowy. Taka metoda będzie jednak niedostateczna przy mierzeniu złożonych pytań z wieloma wątkami, bądź przy takich pytaniach, dla których źródłem może być kilka różnych fragmentów tekstu.

4.2 Aplikacja webowa

Wytworzona została aplikacja webowa do interakcji z systemem RAG, w której użytkownik może porozumiewać się z aplikacją, tak jak z chatbotem. Do wyboru są również 2 typy interfejsów językowych, od których zależy jakość odpowiedzi. Pierwszy implementuje model umiejscowiony na serwerze aplikacji, natomiast drugi jest interfejsem webowym OpenAI łączącym się z nim poprzez REST API.

RAG app

Select LLM

(Local) Flan T5

hello, who are you?

assistant for Cracow University Of Technology

Ask me a question...

Rys. 5 Aplikacja webowa z lokalnym modelem Flan-T5 od Google

RAG app

Select LLM

ChatGPT 3.5

hello, who are you?

I am an assistant for inquiries related to Cracow University of Technology.

Ask me a question...

Rys. 6 Aplikacja webowa z modelem ChatGPT 3.5 od OpenAI

4.3 Dyskusja

Lokalnie zaimplementowany model embedding'owy all-Mini-LM-L6-v2 autorstwa sentence-transformers oraz model językowy Flan-T5 (248M parametrów), który został stworzony przez zespół Google oraz stanowi lepszą wersję wcześniejszego modelu T5 (Text to Text Transfer Transformer) opartego o architekturę transformerową. Jak twierdzą sami autorzy: „, FLAN-T5 is just better at everything. For the same number of parameters, these models have been fine-tuned on more than 1000 additional tasks covering also more languages.”. Jest to model trenowany bezpośrednio pod generowanie tekstu z zadanego prompta, zatem można uznać, że jest modelem konwersacyjnym. Mimo to Flan-T5 nie jest przystosowany do odpowiadania na pytania w sposób przyjazny dla człowieka, stąd też jego odpowiedzi są raczej krótkie i zwięzłe. Porównując go do modelu ChatGPT3.5 widzimy różnicę w czytelności odpowiedzi, za którą odpowiedzialny jest komponent RLHF. Jest to dodatkowy model, oparty o uczenie ze wzmocnieniem, który „nakierowuje” model językowy do generowania odpowiedzi udzielanych pełnymi zdaniami, imitującymi rozmowę z człowiekiem.

5 Podsumowanie

System RAG jest aktualnie jednym z najlepiej rokujących kierunków w rozwoju systemów opartych o modele językowe. Prawdopodobnymi konsekwencjami jego komercyjnego zastosowania mogą być redukcje kosztów oraz czasu wyszukiwania informacji w zbiorach danych. Niemniej jednak system wymaga wzmocnienia jego wiarygodności ze względu na wysokie ryzyko pomyłki przy wyszukiwaniu źródeł, z których następnie korzysta model językowy. Dodatkowo inżynier AI ma niepełną kontrolę nad generowaniem tekstu, co może skutkować generowaniem szkodliwych treści oraz halucynacjami, bądź korzystaniem z „wiedzy” LLMów, która może nie pokrywać się z wiedzą z dokumentów.

6 Bibliografia

- <https://medium.com/@kanavanand8/solving-insurance-doubts-using-the-rag-model-with-llm-in-local-911290fa9889>
- <https://huggingface.co/blog/rlhf>
- <https://betterprogramming.pub/llamaindex-how-to-evaluate-your-rag-retrieval-augmented-generation-applications-2c83490f489>
- <https://medium.com/@rishabhnmije123/langchain-a-powerful-tool-for-local-llm-execution-441df81e2ab>
- <https://huggingface.co/google/flan-t5-base>
- https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2?source=post_page-----911290fa9889-----