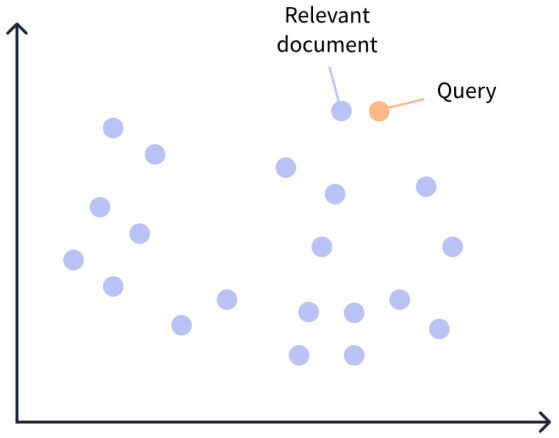
Zadanie 3: Wyszukiwanie semantyczne - wersja 1

Opracowanie: dr inż. Arkadiusz Janz, mgr inż. Konrad Wojtasik

Celem niniejszego ćwiczenia jest zapoznanie się z technologią semantycznego wyszukiwania opartą na modelach językowych. Zadanie wyszukiwania polega na zwróceniu dokumentów, które są relewantne do zapytania użytkownika. Semantyczne wyszukiwanie odnosi się do procesu przetwarzania języka naturalnego (NLP), w którym kluczowym krokiem jest opisanie znaczenia zapytania i znaczenia analizowanych dokumentów, a następnie wykorzystanie uzyskanych opisów semantycznych w dopasowaniu dokumentów do zapytania.



Źródło: https://huggingface.co/learn/nlp-course/chapter5/6

Przykład 1: Zapytanie "Co to jest sztuka?"

- Dokument 1: "Sztuka to forma wyrazu artystycznego."
- Dokument 2: "Sztuka to sposób bycia."

W semantycznym wyszukiwaniu system identyfikuje znaczenie słowa "sztuka" w obu dokumentach i ocenia, że są one relewantne do zapytania.

Przykład 2: Zapytanie "Jaka jest cena produktu Y?"

- Dokument 1: **"Produkt X kosztuje 100 złotych, zaś Produkt Z - 200 złotych."**
- Dokument 2: **"Produkt Y kosztuje 20 złotych."**

W semantycznym wyszukiwaniu system rozpoznaje znaczenie zapytania i potrafi wyodrębnić poprawny dokument, który odpowiada na zapytanie użytkownika.

Współczesne systemy wyszukiwania nie posługują się słowami kluczowymi, gdzie uwzględnia się tylko dosłowne wystąpienia słów czy fraz, a wykorzystują modele przestrzeni wektorowych (Word Embeddings, TF-IDF, LSA) i modele językowe (BERT, RoBERTa, LaBSE i inne) w celu uchwycenia semantyki tekstu. W niniejszym ćwiczeniu rozważone zostaną różnorodne metody osadzania tekstów na potrzeby wyszukiwania semantycznego.

Część I: Polyencoders (7 pkt)

A. Należy zapoznać się z biblioteką *SentenceTransformers* (https://sbert.net/) oraz poniższym opisem
[0.5 pkt]:
https://sbert.net/examples/applications/retrieve_rerank/README.html
B. Należy zapoznać się z podanym artykułem
[0.5 pkt]:
https://arxiv.org/pdf/1905.01969
C. Należy zapoznać się z metrykami oceny opisanymi w podanym artykule
[0.5pkt]:
https://www.pinecone.io/learn/offline-evaluation/

Plan realizacji ćwiczenia

Celem niniejszej części jest weryfikacja działania pre-trenowanych modeli bi-enkoder i cross-enkoder na zadanym zbiorze danych. Należy zaimplementować typowy potok wyszukiwania semantycznego oparty na mechanizmie *retrievera* i *re-rankera*.

1. Należy zapoznać się z opisem typowego potoku wyszukiwania przedstawionym w następującym artykule **[0.5 pkt]**:
https://www.pinecone.io/learn/series/rag/rerankers/

2. Należy zapoznać się z niniejszym tutorialem i wykonać go samodzielnie w ramach własnego środowiska uruchomieniowego **[1 pkt]**:
https://huggingface.co/learn/nlp-course/chapter5/6
    a. należy wykonać kilka przykładowych wyszukiwań posługując się kodem załączonym w tutorialu

b. przeanalizować uzyskane rezultaty i podsumować je; przeanalizować błędy wyszukiwania

3. Analiza działania modelu *retriever* (bi-enkoder) na wybranym zbiorze danych (np. `sentence-transformers/squad` dostępny w ramach platformy HuggingFace **[2.5 pkt]** https://huggingface.co/datasets/sentence-transformers/squad)

   a. sugeruje się wykorzystanie modelu `multi-qa-mpnet-base-dot-v1` (https://sbert.net/examples/applications/retrieve_rerank/README.html)
   b. wykonać osadzenia zapytań i dokumentów (kolumny `question` i `answer`) oraz przeprowadzić analizę przykładowych wyszukiwań, tym razem posługując się innym tutorialem: https://sbert.net/examples/applications/semantic-search/README.html
   c. należy dodać indeks wyszukiwania FAISS w celu dopasowania dokumentów do zapytań za pomocą osadzeń i ponownie wykonać wyszukiwanie (ustalić top_k = 5)
   d. zaimplementować wybraną metrykę oceny skuteczności wyszukiwania
   e. porównać wyniki uzyskane bez wykorzystania indeksu i wraz z wykorzystaniem indeksu FAISS.

4. Analiza działania modelu *re-ranker* (cross-enkoder) **[1.5 pkt]**
   a. do potoku wyszukiwania zaimplementowanego w podpunkcie 2 należy dodać moduł *re-rankingu*
   b. ponownie wykonać wyszukiwania (top_k = 5) dla wybranych zapytań ze zbioru i dokonać *re-rankingu* za pomocą cross-enkodera (proszę wykorzystać jeden z pre-trenowanych modeli: https://sbert.net/docs/pretrained-models/ce-msmarco.html)
   c. ponownie ocenić skuteczność wyszukiwania za pomocą ustalonych metryk oceny i porównać z poprzednio uzyskanymi wynikami.

Część II RAG (3 pkt)

A. Należy zapoznać się z podanym artykułem: https://medium.com/@dinabavli/rag-basics-basic-implementation-of-retrieval-augmented-generation-rag-e80e0791159d

Plan realizacji ćwiczenia

1. Wykorzystać API REST z dostępem do modeli językowych CLARIN
    a. należy zarejestrować się, a następnie zalogować na platformie usług https://services.clarin-pl.eu/login
    b. endpoint z listą modeli: `/api/v1/oapi/models`
    c. endpoint do uruchomienia modelu: `/api/v1/oapi/chat/completions`

2. Należy zaindeksować Wikipedię lub użyć gotowych indeksów FAISS. Sugerowany indeks: https://www.kaggle.com/datasets/jjinho/wikipedia-2023-07-faiss-index

3. Dodać do potoku moduł generacji odpowiedzi (RAG) wykorzystując podane API https://python.langchain.com/docs/tutorials/rag/
    a. przygotować odpowiedni prompt do modelu, który przekształci wyszukaną przez potok dokument na odpowiedź w języku naturalnym
    b. wykonać własne, przykładowe zapytania i przekierować wyniki wyszukiwania na zaimplementowany moduł RAG
    c. przeanalizować rezultaty wyszukiwania po zastosowaniu RAG
    d. podsumować uzyskane wyniki