Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH

Instytut Informatyki

Sprawozdanie z realizacji projektu

Budowa własnego mini-RAG-a z fine-tuningiem retrievera.

Marcin Bagnowski
Numer albumu 325143

WARSZAWA 2025

Spis treści

1. Wstęp

Projekt został zainspirowany pracą: **Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks** [1]

Głównym celem jest implementacja modelu **Retrieval-Augmented Generation (RAG)**, opisanego we wspomnianej publikacji. Model RAG udziela odpowiedzi na pytania w sposób zbliżony do klasycznego modelu językowego (LLM). Jednak różni się źródłem wiedzy wykorzystywanej przy generowaniu odpowiedzi. W przypadku LLM jest to głównie zbiór danych, na którym model był trenowany. W modelu RAG natomiast, odpowiedzi opierają się na zewnętrznych źródłach, takich jak dokumenty, fragmenty tekstu czy całe bazy danych.

Model RAG składa się z dwóch głównych komponentów: **Retrievera**, który wyszukuje fragmenty najbardziej powiązane z zadanym pytaniem, oraz **Generatora** – klasycznego modelu językowego, który generuje odpowiedź, wykorzystując jako kontekst zarówno pytanie, jak i odnalezione fragmenty. W ramach projektu przeprowadziłem dostrajanie (ang. *fine-tuning*) komponentu retrievera i połączyłem go z gotowym generatorem.

Motywacja - Tworzenie modeli typu RAG jest uzasadnione rosnącą potrzebą na systemy zdolne do udzielania trafnych odpowiedzi w zadaniach wymagających dostępu do dużych zasobów wiedzy. Klasyczne modele językowe, mimo wielu zalet, mają ograniczoną zdolność aktualizacji wiedzy po zakończeniu treningu. Model RAG jest rozwiązaniem tego problemu, ponieważ łączy możliwości językowe modeli LLM z dostępem do zewnętrznych źródeł informacji. To czyni go bardziej elastycznym i użytecznym w zastosowaniach, takich jak systemy QA, chatboty czy asystenci wspierający podejmowanie decyzji. Dodatkowo, rozwój takich rozwiązań wspiera tworzenie bardziej przejrzystych i możliwych do weryfikacji modeli, co ma szczególne znaczenie w kontekście rosnących oczekiwań względem zaufania do systemów opartych na sztucznej inteligencji.

1.1. Przykładowe działanie modelu

Stworzony model RAG przyjmuje od użytkownika pytanie w formie tekstu i zwraca: treść fragmentów dokumentów, na podstawie których zbudował swoją odpowiedź oraz udzieloną odpowiedź. Dla przejrzystości przykładów, teksty znalezionych pasaży zostały skrócone.

Question: When did the French Revolution start?

Retrieved passages:

```
   The French Revolution (French: Révolution française) was a period
of political and societal change in France which began with the
Estates General of 1789 and ended with the Coup of 18 Brumaire
on 9 November 1799. Many of the revolution's ideas are considered
fundamental principles...
   === Revolutionary France (1789-1799) === The French Revolution
was a period of political and societal change that began with the
Estates General of 1789, and ended with the coup of 18 Brumaire in
1799 and the formation of the French Consulate...
   === France === The long-term impact on France was profound,
shaping politics, society, religion and ideas, and polarising
politics for more than a century. Historian François Aulard
writes:"From the social point of view, the Revolution...
```

Response: The French Revolution started in May 1789 when the representatives of the Third Estate broke away and re-constituted themselves as a National Assembly in June. This marked the beginning of the revolution and the convocation of the Estates General, which was converted into a National Assembly in June.

Znalezione fragmenty można również wyświetlić z większą liczbą informacji - nie tylko tekst, ale również:

- ranga dokumentu - numer istotności dokumentu (numeracja od najistotniejszego)
- score - wynik istotności dokumentu z pytaniem
- ID pasażu - identyfikator pasażu z korpusu dokumentów
- tytuł dokumentu (artykułu)

Question: What is the main export product of Japan according to the provided texts?

Retrieved passages:

```
   (Rank: 1) (PassageID: wiki_938_chunk_29) (Score: 0.8471) (Title:
Japan) "Japan has the world's fifth-largest economy by nominal GDP,
after...
   (Rank: 2) (PassageID: wiki_678_chunk_20) (Score: 0.8077) (Title:
Asia) "In the late 1980s and early 1990s, Japan's GDP by currency
exchange...
   (Rank: 3) (PassageID: wiki_134_chunk_1) (Score: 0.7918) (Title:
Economy of China) "China is the world's largest manufacturing
industrial economy and exporter of goods...
```

Response: Japan's main export product according to the provided texts is motor vehicles, iron and steel products, semiconductors, and auto parts.

Na pytania, na które odpowiedź nie jest zawarta w korpusie dokumentów, a tym samym nie znajdzie się ona w znalezionych fragmentach dokumentów, model odpowiada, że nie otrzymał koniecznych informacji, aby dopowiedzieć na to pytanie.

Question: Who is the current president of Mars based on the documents?

Retrieved passages:

```
   === Pacifism === On 1 December 1948, Costa Rica abolished
its military force. In 1949, the abolition of the military was
introduced...
   === Foreign relations === The foreign relations of Mexico
are directed by the President of Mexico and managed through the
Ministry of Foreign Affairs...
   == Presidency (1977-1981) == Carter was inaugurated as the 39th
president on January 20, 1977. One of Carter's first acts was the
fulfillment of...
```

Response: Sorry, I don't know the answer based on the articles provided.

1.2. Zrealizowany zakres

W ramach projektu zrealizowano:

1. Identyfikacja i znalezienie niezbędnych zasobów do realizacji projektu: literatury, zbiorów danych, modeli.
2. Przygotowanie zbioru danych do treningu retrievera:
   a) Ekstrakcja kluczowych encji z pytań ze zbioru NQ [2].
   b) Na podstawie najczęściej występujących encji – selekcja około 1000 powiązanych artykułów z anglojęzycznej Wikipedii, tworzących korpus dokumentów.
   c) Podział pobranych artykułów na mniejsze fragmenty - pasaże.
   d) Wyliczenie osadzeń (ang. *embeddings*) wszystkich dokumentów z utworzonego korpusu oraz embeddingów pytań i odpowiedzi ze zbioru NQ.
   e) Wyszukanie top-k najbardziej dopasowanych dokumentów dla każdego pytania z wykorzystaniem gotowego retrievera.
   f) Zbudowanie zbioru: pytanie, dokument, etykieta (dokument związany / nie związany z pytaniem) do fine-tuningu retrievera.
3. Fine-tuning retrievera wykorzystując przygotowany zbiór (pytanie, dokument, etykieta).

4. Pobranie i lokalne uruchomienie generatora.
5. Połączenie retrievera z generatorem, tak aby generator odpowiadał na zadane pytanie, wykorzystując dokumenty/treść dostarczoną przez retriever.

Szczegółowy opis powyższych punktów znajduje się w sekcji 3.4.

2. Przegląd literatury

Z pracy [1] zaczerpnąłem fundamentalną koncepcję dwuetapowego procesu: najpierw retriever wyszukuje relevantne dokumenty, a następnie te dokumenty, wraz z oryginalnym zapytaniem, są przekazywane do modelu generatora. W tym projekcie, podobnie jak w pracy [1] tworzymy taki system, aczkolwiek skupiamy się na fine-tuningu wyłącznie komponentu retrievera. Korzystamy z gotowego modelu generatora.

Efektywność retrievera jest krytyczna dla całego systemu RAG. Praca [3] zademonstrowała modele Dense Passage Retriever (DPR), które wykorzystują embeddingi do reprezentacji zapytań i dokumentów. Chociaż nie implementuję DPR od zera, to korzystam z idei wykorzystania gęstych reprezentacji (embeddingów) do oceny podobieństwa między zapytaniem a fragmentami dokumentów. Co więcej, strategia identyfikacji "trudnych negatywów"(wysokie podobieństwo do pytania, niskie do odpowiedzi) jest inspirowana metodami opisanymi w artykule [3].

Prace nad Sentence-BERT [4] i rozwój biblioteki **sentence-transformers** umożliwiły łatwy fine-tuning modeli bi-enkoderowych [5] - tutaj modelu (BAAI/bge-base-en-v1.5) [6] jako bazę do fine-tuningu. Do stworzenia własnego zbioru treningowego wykorzystałem idee modelu-nauczyciela: silniejszy model (BAAI/bge-large-en-v1.5) [7] powiązał pary (pytanie, dokument) z mojego korpusu Wikipedii i pytań NQ.

Jako generator wybrałem Qwen/Qwen1.5-1.8B-Chat [8] ze względu na jego dobre wyniki (przy nie tak dużym rozmiarze) oraz za wyjątkowo duże okno kontekstu 32K tokenów, co jest istotne dla systemów RAG. Generator nie był dodatkowo trenowany na nowym zbiorze danych, jak w przypadku retrievera, natomiast został on dostrojony techniką instruction-prompt [9]. Przed dołączeniem znalezionych dokumentów oraz faktycznego pytania model otrzymuje tekst, aby odpowiadał tylko na podstawie dostarczonych dokumentów oraz aby odpowiedział, że nie zna odpowiedzi na pytanie, jeżeli brakuje mu informacji w otrzymanych dokumentach. Oto zastosowana instrukcja:

> Answer the given question USING ONLY information in provided contexts. If the exact answer doesn't appear in provided context, then EXACTLY THIS: 'Sorry, I don't know the answer based on the articles provided.' and don't say anything after that

3. Opis rozwiązania

3.1. Technologie, narzędzia, techniki

1. **Podstawy i Zarządzanie Projektem:**
   - Python 3.12.3
   - Poetry: Wirtualne środowisko, zarządzanie zależnościami.
   - Jupyter Notebook: Interaktywne środowisko deweloperskie.
2. **Przetwarzanie Danych i Przygotowanie Korpusu:**
   - Datasets (Hugging Face): Biblioteka do załadowania zbioru Natural Questions.
   - spaCy: Biblioteka do przetwarzania języka naturalnego, użyta do ekstrakcji encji nazwanych (Named Entity Recognition - NER) z pytań, co było kluczowe dla selekcji artykułów do korpusu wiedzy.
   - wikipedia: API Wikipedii, umożliwiająca pobieranie treści artykułów na podstawie zidentyfikowanych encji.
   - Langchain (langchain.text_splitter): Narzędzie do efektywnego dzielenia długich artykułów z Wikipedii na mniejsze, semantycznie spójne fragmenty (pasaże).
   - NumPy: Biblioteka do obliczeń numerycznych.
3. **Budowa i Trening Modeli (Retriever i Generator):**
   - PyTorch: Główny framework do głębokiego uczenia, wsparcie dla akceleracji GPU (CUDA).
   - Transformers (Hugging Face): Biblioteka dostarczająca pre-trenowane modele oraz narzędzia do ich fine-tuningu i inferencji. Użyta m. in. do załadowania modelu generatora [8] oraz zdefiniowania pętli treningowej do fine-tuningu retrievera [6].
   - Sentence-Transformers (Hugging Face): Biblioteka do tworzenia embeddingów zdań i tekstu. Użyta do załadowania modelów retrieverów [6] [7], fine-tuningu i ewaluacji.
   - scikit-learn: Biblioteka dostarczająca metryki do ewaluacji modelu.
   - BitsAndBytes: Biblioteka umożliwiająca kwantyzację modeli, model generatora został skwantowany do 8-bitów, co znacząco zredukowało zużycie pamięci VRAM.
4. **Monitorowanie i Wizualizacja:**
   - Matplotlib: Użyta do stworzenia wykresów funkcji straty i metryk ewaluacyjnych podczas treningu.
5. **Techniki**
   - Dense Retrieval: Wykorzystanie gęstych reprezentacji wektorowych (embeddingów) do wyszukiwania semantycznie podobnych fragmentów tekstu.

- Wykorzystanie większego, bardziej wydajnego modelu [7] do tworzenia etykiet dla zbioru treningowego mniejszego modelu retrievera.
- Hard Prompting / Instruction Prompting: Dodanie precyzyjnej instrukcji tekstowej dla modelu generatora, aby kontrolować jego zachowanie.
- Kwantyzacja modelu: Zastosowanie techniki kwantyzacji (8-bit) w celu zmniejszenia wymagań pamięciowych modelu generatora.

3.2. Zbiory danych

1. **Zbiór QA:** *Natural Questions* [2] – zestaw około 100 tysięcy pytań i odpowiedzi pochodzących z rzeczywistych zapytań użytkowników Google. Na potrzeby projektu, ograniczyłem zbiór do 1000 przykładowych par pytanie–odpowiedź w celu zmniejszenia kosztów obliczeniowych i skrócenia czasu trenowania.
2. **Korpus dokumentów:** *mini Wikipedia dump* – ograniczony podzbiór artykułów z anglojęzycznej Wikipedii. Wyselekcjonowałem 1000 artykułów, które tematycznie powiązane są z pytaniami ze zbioru QA. Selekcja została przeprowadzona poprzez ekstrakcję kluczowych encji z pytań. Encje posłużyły do odnalezienia odpowiednich artykułów, które utworzyły końcowy korpus wiedzy wykorzystywany przez komponent Retrievera. Następnie artykuły zostały podzielone na mniejsze, semantycznie spójne części. Powstały zbiór składa się z ID pasażu, ID dokumentu, tytułu, tekstu, numerem części dokumentu.
3. **Zbiór do fine-tuningu**: (Pytanie, Dokument, Etykieta) Powstał w wyniku połączenia poprzednich dwóch zbiorów z wykorzystaniem większego retrievera do powiązania pytania z właściwym dokumentem, porównując wskazane dokumenty z odpowiedzią na pytanie, ustalana była etykieta, czy dokument jest czy nie jest związany.

3.3. Modele

- **Retriver 1:** [7] do przygotowania zbioru treningowego, powiązanie par pytanie-dokument
  - nazwa: BAAI/bge-large-en-v1.5
  - liczba parametrów: 335M
  - rozmiar: 1.35 GB (z precyzją float 32)
  - maksymalne okno kontekstu: 512 tokenów
- **Retriver 2:** [6] faktyczny retriever powstałego systemu RAG, na nim był wykonywany fine-tuning.
  - nazwa: BAAI/bge-base-en-v1.5
  - liczba parametrów: 110M
  - rozmiar: 220M (z precyzją float 16)
  - maksymalne okno kontekstu: 512 tokenów

  Zastanawiałem się nad zastosowaniem wersji 'small' zamiast 'base', ale z racji możliwości sprzętowych oraz zwracając uwagę na różnice jakościowe obu modeli 3.1

zdecydowałem się na ten większy. Warty uwagi jest fakt, że przeskok między wersją 'base' oraz 'large' nie jest już tak znaczący.

Table 2: Performance of various models on C-MTEB.

model	Dim	Retrieval	STS	Pair CLF	CLF	Re-rank	Cluster	Average
Text2Vec (base)	768	38.79	43.41	67.41	62.19	49.45	37.66	48.59
Text2Vec (large)	1024	41.94	44.97	70.86	60.66	49.16	30.02	48.56
Luotuo (large)	1024	44.40	42.79	66.62	61.0	49.25	44.39	50.12
M3E (base)	768	56.91	50.47	63.99	67.52	59.34	47.68	57.79
M3E (large)	1024	54.75	50.42	64.30	68.20	59.66	**48.88**	57.66
Multi. E5 (base)	768	61.63	46.49	67.07	65.35	54.35	40.68	56.21
Multi. E5 (large)	1024	63.66	48.44	69.89	67.34	56.00	48.23	58.84
OpenAI-Ada-002	1536	52.00	43.35	69.56	64.31	54.28	45.68	53.02
BGE (small)	512	63.07	49.45	70.35	63.64	61.48	45.09	58.28
BGE (base)	768	69.53	54.12	77.50	67.07	64.91	47.63	62.80
BGE (large)	1024	**71.53**	**54.98**	**78.94**	**68.32**	**65.11**	48.39	**63.96**

Rysunek 3.1. Porównanie jakości wersji modeli BGE (i innych) w zależności od ich rozmiaru [5].

- **Generator:** [8] tworzy finalną odpowiedź na podstawie zadanego pytania i dokumentów dostarczonych przez retriever. Dodatkowo został dostrojony przy pomocy instrukcji 2 dołączanej do każdego zapytania.
  - — nazwa: Qwen/Qwen1.5-1.8B-Chat
  - — parametry: 1.8B
  - — rozmiar: 1.8 GB (z kwantyzacją 8-bit)
  - — maksymalne okno kontekstu: 32 768 tokenów

3.4. Szczegółowy przebieg prac

3.4.1. Identyfikacja i znalezienie niezbędnych zasobów do realizacji projektu

: Przegląd i zapoznanie się z literaturą, w szczególności z pracą [1] definiującą architekturę RAG. Następnie wybór kluczowych elementów projektu, tj. zbiorów danych oraz modeli.

3.4.2. Przygotowanie zbioru danych do treningu retrievera

Ten etap obejmował szereg kroków mających na celu stworzenie wysokiej jakości zbioru par (pytanie, fragment dokumentu, etykieta) dla procesu fine-tuningu retrievera.

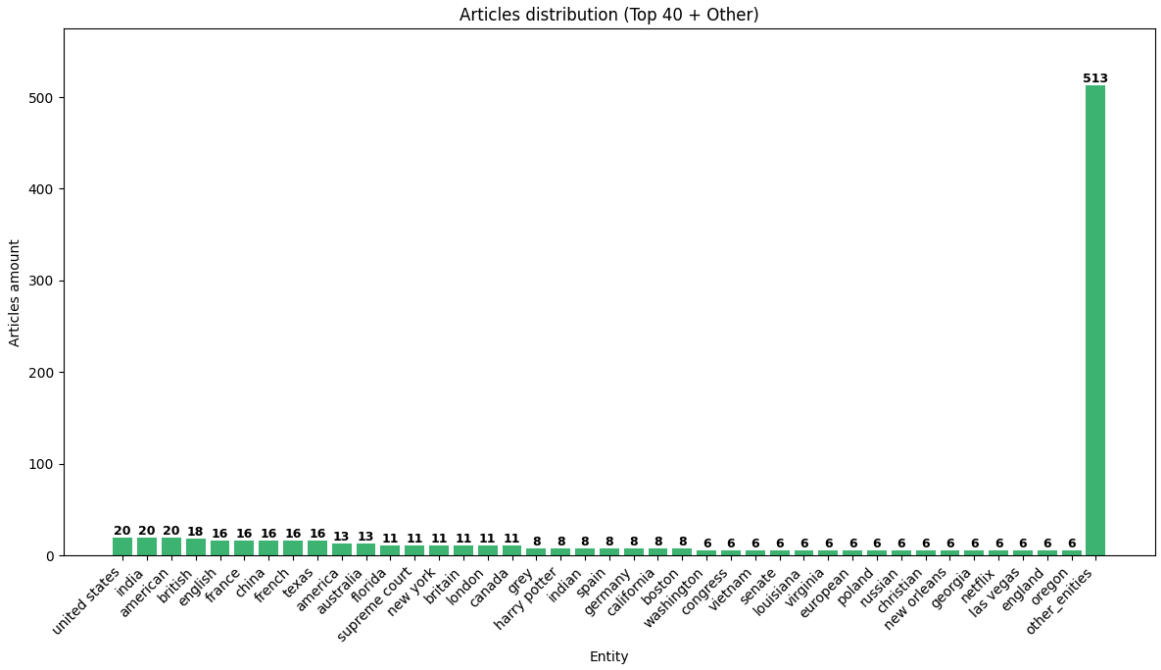
Ekstrakcja kluczowych encji z pytań NQ:

- Pobrano 1000 losowych przykładów ze zbioru NQ.
- Dla każdego z nich, przy użyciu biblioteki spaCy, przeprowadzono ekstrakcję encji nazwanych (NER). Uwzględniono encje typów: PERSON, NORP, FAC, ORG, GPE, LOC, PRODUCT, EVENT, WORK_OF_ART, LAW, LANGUAGE.

- Wyekstrahowane encje poddano prostemu pre-processingowi (usunięcie przedrostka "the "i przyrostka ”s"). W ten sposób uzyskałem **207 unikalnych encji** (łącznie 324 z powtórzeniami).

Selekcja artykułów z Wikipedii i tworzenie korpusu:

- Zliczono częstości występowania wszystkich wyekstrahowanych encji.
- Dla tych encji, pobrano artykuły z anglojęzycznej Wikipedii. Zastosowano strategię dystrybucji, która przypisywała każdej z topowych encji co najmniej **1** artykuł, a maksymalnie **20** artykułów, starając się osiągnąć docelową liczbę około **1000 artykułów**. Rozkład liczby artykułów, dla najczęstszych encji prezentuje wykres 3.2.



Rysunek 3.2. Dystrybucja liczby artykułów dla najczęściej występujących encji.

Podział artykułów na mniejsze fragmenty (pasaże):

- Każdy pobrany artykuł z Wikipedii został podzielony na mniejsze, częściowo nakładające się fragmenty (chunks/pasaże) przy użyciu RecursiveCharacterTextSplitter z biblioteki Langchain.
- Jako funkcję liczenia długości wykorzystano tokenizator docelowego modelu retrievera (BAAI/bge-base-en-v1.5) [6]. Ustawiono docelową długość fragmentu na 320 tokenów i nakładanie się fragmentów na 64 tokeny. Każdy fragment otrzymał unikalne ID i został zapisany wraz z ID i tytułem oryginalnego dokumentu. To dało łącznie **35608 pasaży**.

Wyliczenie osadzeń (embeddingów):

- Przy użyciu większego modelu (BAAI/bge-large-en-v1.5)[7], wygenerowano embeddingi dla:
  — Wszystkich pytań i odpowiedzi ze zbioru NQ.
  — Wszystkich utworzonych fragmentów (pasaży) z korpusu Wikipedii.
- Embeddingi korpusu zostały zapisane lokalnie w celu uniknięcia ich ponownego obliczania.

Wyszukiwanie dopasowanych dokumentów (generowanie kandydatów):

- Dla każdego pytania NQ, na podstawie podobieństwa kosinusowego jego embeddingu do embeddingów wszystkich fragmentów korpusu, wyszukano **50** najbardziej podobnych pasaży.

Budowanie zbioru treningowego (pytanie, dokument, etykieta):

- Dla każdego pytania i jego puli kandydatów nadano etykietę 0/1:
  — Obliczono podobieństwo kosinusowe każdego fragmentu-kandydata do embeddingu odpowiedzi na dane pytanie NQ.
  — **Pozytywne przykłady** (etykieta 1): Wybrano do **dwóch** fragmentów, które miały wynik najwyższe podobieństwa do odpowiedzi, minimum **0.45**. Zastosowano również strategię "ratunkowego"dodawania jednego pozytywnego przykładu dla pytań, które nie znalazły żadnego pozytywnego fragmentu - próg był obniżany do **0.25**. Testowałem również wyższe wartości progu podobieństwa, jednak wtedy wiele pytań nie otrzymywało żadnych fragmentów pozytywnych.
  — **Trudne negatywy** (etykieta 0): Wybrano **3** fragmenty, które były podobne do pytania, ale miały niską zgodność z odpowiedzią. Sortowano je tak, aby wybrać te najbardziej "zwodnicze"(wysokie podobieństwo do pytania, niskie do odpowiedzi).
  — **Łatwe negatywy** (etykieta 0): Dodano **1** losowy fragment z korpusu, który nie był wśród topowych kandydatów dla danego pytania.
- W ten sposób powstał finalny zbiór danych treningowych, liczący **4700 przykładów**, w tym **1988 pozytywnych** i **2702 negatywnych**. Następnie podzielono go na zbiory: treningowy (80%), walidacyjny (10%) i testowy (10%).

3.4.3. Fine-tuning retrievera

- Jako funkcję straty zastosowano losses.CosineSimilarityLoss, która dąży do maksymalizacji podobieństwa kosinusowego dla par pozytywnych (blisko 1.0) i minimalizacji dla par negatywnych (blisko -1.0, po transformacji etykiety 0 na -1).
- Trening przeprowadzono przy użyciu transformers.Trainer, co pozwoliło na bardziej szczegółowe monitorowanie i ewaluację. Parametry treningu obejmowały: *liczba epok: 5, batch size: 16, stopa uczenia: 2* $*10^{-5}$. Zastosowano strategię zapisu najlep-

szego modelu na podstawie metryki f1-score. To nastąpiło dla 450 kroku. Dokładne omówienie wyników fine-tunigu jest w sekcji 4.

3.4.4. Przygotowanie Generatora

- Model generatora Qwen/Qwen1.5-1.8B-Chat [8] i jego tokenizer został pobrany z Hugging Face Hub i zapisany lokalnie.
- Zastosowano kwantyzację 8-bitową przy użyciu biblioteki bitsandbytes w celu zmniejszenia zużycia pamięci VRAM i umożliwienia lokalnego uruchomienia.
- Przeprowadzono wstępne testy generatora na przykładowych promptach, aby zweryfikować jego zdolności generatywne i poprawne formatowanie wejścia/wyjścia.

Prompt: What is the capital of France?

Response: `The capital of France is Paris. It is located in the central-western part of the country and is the largest city in France, with a population...`

Prompt: Write a short story about a friendly robot who discovers a hidden garden.

Response: `Once upon a time, there was a friendly robot named Zora who lived in a vast factory. Zora was programmed to perform various tasks for the...`

Prompt: Translate the following English sentence to Polish: 'Hello, how are you today?'

Response: `Pani, jak jest ci dzisiaj?`

3.4.5. RAG Pipeline

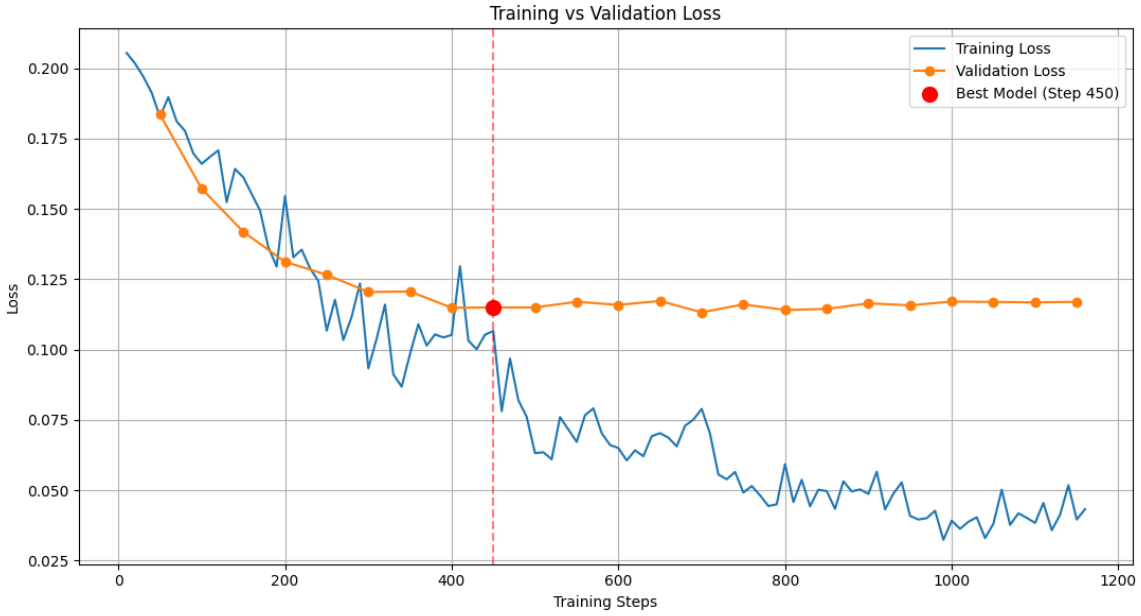
- Stworzono finalny potok RAG, który integruje wytrenowany lokalny retriever oraz przygotowany lokalny generator.
- Przeprowadzono jakościowe testy całego systemu na różnych typach zapytań, w tym takich, na które odpowiedź powinna znajdować się w korpusie, oraz takich, dla których oczekiwano odpowiedzi o braku informacji. Przykładowe zapytania znajdują się w sekcji 1.1.

4. Wyniki ewaluacji eksperymentalnej

4.1. Metryki i ocena jakości Retrievera

Do oceny jakości fine-tuningowanego modelu retrievera [6] wykorzystano zbiór walidacyjny oraz zbiór testowy, wyodrębnione z przygotowanego wcześniej zestawu par (pytanie, fragment dokumentu, etykieta). Ocena odbywała się z użyciem następujących metryk: Accuracy, Precision, Recall, F1-score.

Proces fine-tuning'u modelu retrievera [6] był monitorowany na zbiorze walidacyjnym. Przebieg kluczowych metryk został zaprezentowany na rysunku 4.2, a wykres funkcji straty na rysunku 4.1.



Rysunek 4.1. Przebieg funkcji straty na zbiorach treningowym i walidacyjnym.

Obserwujemy spadek funkcji straty na obu zbiorach, z konwergencją w okolicy 450 kroków. Model z tego stanu został zapisany jako najlepszy.

Metryka F1-score, wybrana jako kryterium wyboru najlepszego modelu, osiągnęła maksymalną wartość **0.854**. W tym samym punkcie, precyzja wynosiła **0.862**, a czułość **0.845**, co wskazuje na zbalansowaną zdolność modelu do identyfikacji relewantnych fragmentów przy jednoczesnym unikaniu fałszywych pozytywów.

Po zakończeniu treningu, najlepszy model został oceniony na odłożonym zbiorze testowym. Uzyskane wyniki przedstawiono w tabeli 4.1.

Analiza wyników retrievera: Na podstawie uzyskanych metryk można stwierdzić, że fine-tuning znacząco poprawił zdolność modelu bazowego do identyfikacji relewantnych fragmentów dla naszego korpusu dokumentów i wybranych pytań NQ. Wysoki F1-score

Rysunek 4.2. Przebieg wartości metryk na zbiorze walidacyjnym.

Tabela 4.1. Wyniki ewaluacji najlepszego modelu retrievera na zbiorze testowym

Metryka	**Wartość**
Accuracy (przy progu 0.5)	0.880
F1-score (przy progu 0.5)	0.858
Precision (przy progu 0.5)	0.871
Recall (przy progu 0.5)	0.845

na zbiorze testowym wynoszący **0.858** sugeruje dobre zbalansowanie między precyzją a czułością.

4.2. Ocena Jakościowa Systemu RAG

Przeprowadziłem ręczną ocenę jakościową całego systemu RAG, zadając przykładowe pytania. Celem było sprawdzenie, czy generator udziela poprawnych odpowiedzi w oparciu o otrzymane fragmenty, a także czy poprawnie reaguje w sytuacjach, gdy odpowiedź nie znajduje się w dostarczonym kontekście. Oto kilka przykładów testowych (plus sekcja 1.1) wraz z oryginalnymi odpowiedziami ze zbioru NQ:

Question: where did the 2002 world cup take place

Retrieved passages:

```
=== First World Cup team === In 1954, South Korea entered FIFA
World Cup qualification for the first time, and qualified for the
1954 FIFA World Cup...
```

```
   England failed to qualify for the World Cup in 1974, 1978
and 1994. The team's earliest exit in the finals tournament was
elimination...
   Peru has taken part in the World Cup finals five times.
The Peruvian team competed at the first World Cup in 1930 by
invitation...
```

Response: The 2002 FIFA World Cup took place in South Korea.

Original NQ answer: 2002 FIFA World Cup The 2002 FIFA World Cup was the 17th FIFA World Cup, the quadrennial world championship for men's national football teams organized by FIFA. It was held from 31 May to 30 June 2002 at sites in South Korea and Japan, with its final match hosted by Japan at International Stadium in Yokohama.

Question: what is the title of the second harry potter book

Retrieved passages:

```
   The second book, Harry Potter and the Chamber of Secrets, was
originally published in the UK on 2 July 1998 and in the US on 2
June 1999... == Legacy == === Sequels === The second book, Harry
Potter and the Chamber of Secrets, was originally published in the
UK... == Background == === Series === The first novel in the Harry
Potter series, Harry Potter and the Philosopher's Stone...
```

Response: The title of the second Harry Potter book is Harry Potter and the Chamber of Secrets.

Original NQ answer: Harry Potter and the Chamber of Secrets Harry Potter and the Chamber of Secrets is a fantasy novel written by British author J. K. Rowling and the second novel in the Harry Potter series...

Udało mi się też znaleźć przykład ze zbioru 1000 pobranych pytań NQ dla którego model nie zna odpowiedzi.

Question: where did the battle of lake george take place

Retrieved passages:

```
   === Battle of New Orleans === During the final campaign of the
War of 1812, the British sent a force of 11,000 in an attempt to
capture New Orleans...
```

```
   Both sides tried to invade the other and were repulsed. The
American militia proved ineffective because the soldiers were
reluctant to leave home...
   The Battle of New Orleans was fought on January 8, 1815, between
the British Army under Major General Sir Edward Pakenham and the
United States Army under...
```

Response: Sorry, I don't know the answer based on the articles provided.

Original NQ answer: Battle of Lake George The Battle of Lake George was fought on 8 September 1755, in the north of the Province of New York...

System RAG dobrze wykorzystuje pobrane przez retrievera fragmenty do sformułowania trafnych odpowiedzi. Generator poprawnie trzymał się instrukcji, aby opierać się wyłącznie na dostarczonym kontekście, jawnie odpowiadał, że nie zna odpowiedzi, kiedy nie otrzymywał właściwych informacji. Problemem okazuje się zbyt mały korpus wiedzy w stosunku do zróżnicowania pytań w zbiorze NQ, gdyż na część pytań nie ma odpowiedzi w korpusie dokumentów. W efekcie retriever nie zawsze dostarcza dokumenty z odpowiedzią. Należałoby albo zwiększyć korpus dokumentów, albo zawęzić tematykę pytań w zbiorze. Problem błędnie dostarczanych dokumentów raczej też nie wynika ze słabości retrievera, patrząc na uzyskane wartości metryk. Przyczyną jest raczej za mało mocnych przykładów pozytywnych w zbiorze treningowym. To można naprawić, stosując lepszy model do wyliczania embeddingów i poprzez zawężenie tematyczne pytań.

5. Podsumowanie

Niniejszy projekt miał na celu zaprojektowanie, implementację oraz wstępną ewaluację systemu Retrieval-Augmented Generation (RAG) ze szczególnym naciskiem na proces fine-tuning'u komponentu retrievera. Głównym założeniem było stworzenie funkcjonalnego mini-RAG, który wykorzystuje ograniczony korpus wiedzy bazujący na artykułach z Wikipedii i jest w stanie generować odpowiedzi na ich podstawie.

5.1. Krytyczna analiza osiągniętych rezultatów:

W ramach projektu pomyślnie zrealizowałem wszystkie kluczowe etapy:

1. Przygotowanie dedykowanego korpusu dokumentów i zbioru treningowego dla retrievera.
2. Fine-tuning retrievera.
3. Integracja z generatorem i stworzenie kompletnego potoku RAG.

Przygotowanie zbioru danych okazało się najbardziej wymagającym etapem. Początkowo dystrybucja artykułów do encji była robiona funkcją soft-max, co bardzo ograniczyło zakres tematyczny korpusu. Wprowadzenie bardziej zrównoważonej dystrybucji artykułów per encja (z minimalną i maksymalną liczbą artykułów) oraz modyfikacje w procesie tworzenia pozytywnych przykładów (w tym strategia "ratunkowego"dodawania pozytywów i obniżenie progu minimalnego podobieństwa) pozwoliły na uzyskanie lepszego zbioru treningowego. Jednak nadal można tu dużo poprawić (więcej w następnej sekcji 5.2).

Fine-tuning retrievera na tym zbiorze, monitorowany za pomocą metryk takich jak F1-score, precyzja i czułość na zbiorze walidacyjnym, wykazał zdolność modelu do nauki rozróżniania relewantnych i nierelewantnych fragmentów dla zadanych pytań w około 88%, co potwierdziły wyniki na zbiorze testowym.

Integracja retrievera z generatorem przebiegła już dosyć łatwo. Do przetestowania był głównie prompt z instrukcją do generatora, chociaż już po chwili jego kalibracji generator zaczął działać w oczekiwany sposób.

5.2. Dyskusja i propozycja dalszych prac:

Pomimo osiągniętych rezultatów, istnieje kilka kierunków, w których opracowane rozwiązanie mogłoby zostać znacząco usprawnione:

- **Udoskonalenie procesu tworzenia zbioru treningowego dla retrievera.** Należałoby zwiększyć korpus dokumentów, tak aby pokrywał on większą tematykę. Można również ograniczyć zestaw pytań do konkretnej tematyki, aby również uprościć dobór dokumentów. Kolejną opcją są modyfikacje ekstrahowania kluczowych encji z pytań i wyszukiwania powiązanych dokumentów. Również zastosowanie lepszego retrievera do etykietowania par (pytanie, dokument) mogło by być pomocne.
- **Wykorzystanie zbioru NQ z większą liczbą odpowiedzi na pytanie.** To pozwoliłoby na lepszą ocenę dokumentów (potencjalnych pozytywnych) do zadanego pytania. Można np. wyciągać średnie podobieństwo znalezionego tekstu z każdą odpowiedzią.
- **Zastosowanie kilku retrieverów oraz rankera.** Zastosowanie kilku retrieverów zwiększyło by dokładność znajdowanych dokumentów, zwłaszcza jeśli korpus zostałby zwiększony. Jako, że każdy retriever zwróciłby uwagę na inne aspekty podobieństwa pytanie-dokument, konieczny były również ranker, który zadecydowały, które dokumenty są faktycznie najlepsze.
- **Strojenie generatora.** Nie jest to tak kluczowe usprawnienie jak wymienione możliwości poprawy, jednak lepszy dobór instrukcji lub zastosowanie innych technik mogłoby poprawić zwięzłość i rzeczowość zwracanych odpowiedzi. Pomocna może być tutaj praca [9].

Podsumowując, zrealizowany projekt stanowi dobrą podstawę i demonstrację możliwości budowy systemu RAG. Zidentyfikowane obszary do dalszych prac otwierają drogę do stworzenia jeszcze bardziej wydajnego i dokładnego rozwiązania.

Repozytorium

Cały kod jest publicznie dostępny na repozytorium https://github.com/Kapturz0ny/rag-wiki.

Sprzęt

Wszystkie eksperymenty zostały przeprowadzone z użyciem GPU RTX 5070ti 16GB VRAM.

Bibliografia

[1] P. Lewis, E. Perez, A. Piktus i in., „Retrieval-augmented generation for knowledge-intensive nlp tasks”, *Advances in neural information processing systems*, t. 33, s. 9459–9474, 2020.

[2] S. Transformers, *Natural Questions Dataset on Hugging Face*. adr.: `https://huggingface.co/datasets/sentence-transformers/natural-questions`.

[3] V. Karpukhin, B. Oguz, S. Min i in., „Dense Passage Retrieval for Open-Domain Question Answering.”, w *EMNLP (1)*, 2020, s. 6769–6781.

[4] N. Reimers i I. Gurevych, „Sentence-bert: Sentence embeddings using siamese bert-networks”, *arXiv preprint arXiv:1908.10084*, 2019.

[5] S. Xiao, Z. Liu, P. Zhang i N. Muennighoff, *C-Pack: Packaged Resources To Advance General Chinese Embedding*, 2023. arXiv: `2309.07597 [cs.CL]`.

[6] S. Xiao, Z. Liu, P. Zhang i N. Muennighoff, *BAAI/bge-base-en-v1.5*. adr.: `https://huggingface.co/BAAI/bge-base-en-v1.5`.

[7] S. Xiao, Z. Liu, P. Zhang i N. Muennighoff, *BAAI/bge-large-en-v1.5*. adr.: `https://huggingface.co/BAAI/bge-large-en-v1.5`.

[8] J. Bai, S. Bai, Y. Chu i in., „Qwen Technical Report”, *arXiv preprint arXiv:2309.16609*, 2023.

[9] J. Wei, M. Bosma, V. Y. Zhao i in., „Finetuned language models are zero-shot learners”, *arXiv preprint arXiv:2109.01652*, 2021.