RAG (Retrieval Augmented Generation) oraz wpływ różnych embeddingów na działanie tej metody.

Ten projekt bada wpływ różnych embeddingów na działanie metody Retrieval Augmented Generation (RAG) na przykładzie bazy wektorowej Chroma.

Cel projektu

Początkowo chcieliśmy porównać kilka bardziej znanych modeli generujących embeddingi. Jednakże po odkryciu benchmarku MTEB (Massive Text Embedding Benchmark) [9] stwierdziliśmy, że lepszym pomysłem byłoby wybranie modelu który nie został jeszcze uwzględniony w tym benchmarku, a następnie wstawienie do niego uzyskanych przez nas wyników. Dzięki temu zamiast liczyć coś co zostało już wcześniej policzone w jakimś stopniu przyczynimy się do rozwoju tego otwartoźródłowego projektu.

Zakres projektu: 1. Analiza podejścia Retrieval Augmented Generation. 2. Zdefiniowanie metod ewaluacji skuteczności działania metody RAG. 3. Zdefiniowanie testowanych embeddingów. 4. Przeprowadzenie testów. 5. Analiza wyników i wnioski.

Czym jest RAG?

Jest to skrót od "Retrieval-Augmented Generation", czyli w tłumaczeniu na polski "Generacja z użyciem mechanizmu odzyskiwania". RAG jest często wykorzystywany w zadaniach związanych z przetwarzaniem języka naturalnego, takich jak systemy odpowiadające na pytania, generatory tekstu czy też w systemach dialogowych. Wdrażanie takiego systemu może przyczynić się do lepszej skuteczności i dostarczenia bardziej trafnych oraz aktualnych odpowiedzi w kontekście danego zadania. [7, 8]

Jak działa RAG?

RAG łączy procesy odzyskiwania informacji (Retrieval) i generacji tekstu (Generation). Działanie RAG można podzielić na kilka kluczowych etapów:

0. **Utworzenie bazy wiedzy**:
    - Podstawą jest utworzenie bazy wiedzy do przeszukiwania w kontekście zapytań użytkownika. W naszym projekcie jej realizacja będzie miała postać bazy wektorowej zawierającej embeddingi pochodzące z dokumentów PDF zapewnionych przez użytkownika.
1. **Zapytanie Użytkownika**:
    - Zapytanie użytkownika jest przekazywane do systemu RAG. Zapytanie to może być pytaniem, prośbą o wygenerowanie tekstu lub jakimkolwiek innym promptem.

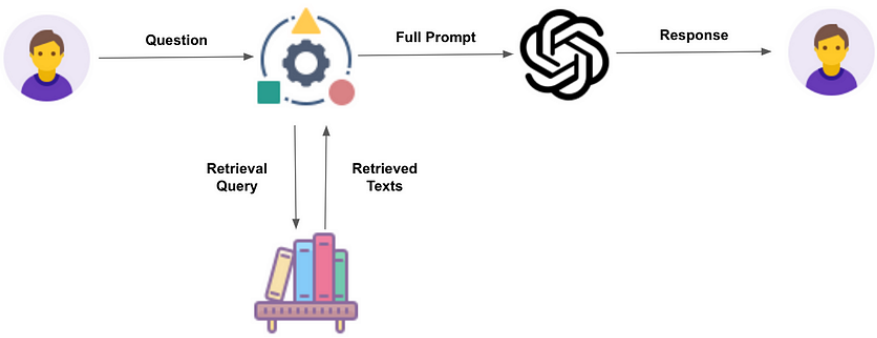

Figure 1: RAG [6]

2. **Odzyskiwanie Informacji (Retrieval)**:
   - Zapytanie użytkownika zostaje zembeddowane do bazy wektorowej. W tym kroku należy pamiętać aby wykorzystać ten sam model, którego wcześniej użyliśmy do utworzenia wektorowej bazy danych.
   - System pobiera k najbardziej podobnych elementów z bazy wektorowej. Do znalezienia najbardziej podobnych elementów pod względem semantycznym wykorzystywana jest najczęściej odległość cosinusowa (cosine similarity).
3. **Generacja Tekstu (Generation)**:
   - Pobrane k najbardziej podobnych elementów jest łączone z zapytaniem użytkownika.
   - Tak przygotowane zapytanie użytkownika wraz z pobranym kontekstem jest wykorzystywane przez model generacji tekstu, aby stworzyć nową, spójną odpowiedź.
   - Ta odpowiedź jest następnie przekazywana użytkownikowi jako końcowy produkt działania systemu RAG.

Jak można ewaluować działanie systemu RAG?

Metodę RAG można ewaluować względem następujących aspektów: - ewaluacja odzyskiwania informacji (tylko Retrieval) - Context Precision (dokładność kontekstu) - Context Recall (pełność kontekstu) - ewaluacja generacji tekstu (tylko Generation) - Generation Faithfullness (wierność generacji) - Generation Relevance (znaczenie generacji)

Chcąc ewaluować RAG całościowo, możemy patrzeć na średnią harmoniczną powyższych czterech metryk. W taki sposób działa metoda **ragas score** [4]. W tym projekcie skupimy się natomiast przede wszystkim na ewaluacji odzyskiwania informacji (Retrieval). Oczywiście zarówno aspekt Retrieval jak i Generation są niezwykle istotne, natomiast w naszym przypadku, tj. w przypadku badania wpływu różnych embeddingów, szczególnie interesującym będzie właśnie aspekt

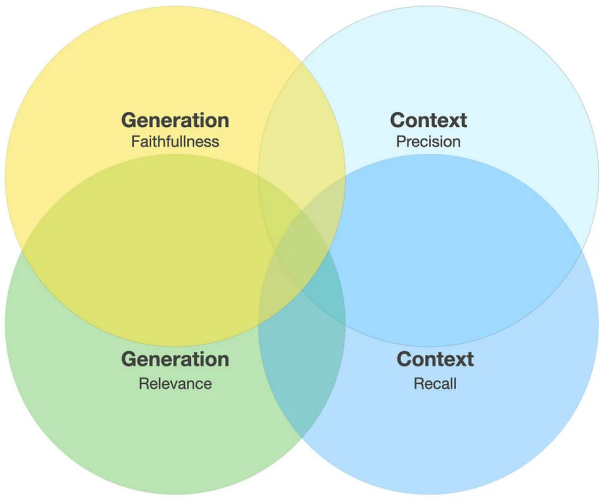

Figure 2: RAG - ewaluacja [5]

Retrieval.

Zarówno Generation Faithfullness (czyli mierzenie halucynacji) jak i Generation Relevance (czyli mierzenie jak dobrze wygenerowana odpowiedź odnosi się do zadanego pytania) zależą przede wszystkim od modelu wykorzystanego do wygenerowania ostatecznej odpowiedzi. Natomiast model tworzący embeddingi będzie miał bezpośredni wpływ na aspekt odzyskiwania informacji (Retrieval). W związku z powyższym wybierając metodę ewaluacji skupiliśmy się tylko i wyłącznie na aspekcie Retrieval.

Wybrana metoda ewaluacji

Dokonujemy badania miarą główną n-DCG@10 z miarami pomocniczymi MRR@k oraz MAP@k na zadaniu Retrieval projektu MTEB [2] na w sumie 15 benchmarkach po raz pierwszy skompilowanych na projekcie BEIR [3]. Do ewaluacji wykorzystywane jest narzędzie mteb, do podglądu rezultatów oraz oceny manualnej używany jest system RAG z wektorową bazą danych Chroma zbudowany w technologii LangChain z interfejsem w Streamlit.

- n-DCG@10 (Normalized Discounted Cumulative Gain at 10)
  - Skupia się na ocenie jakości pierwszych 10 wyników, co jest istotne w kontekście szybkiego dostarczania trafnych odpowiedzi użytkownikom.
- MRR@k (Mean Reciprocal Rank at k)
  - Ocenia, jak dobrze system odzyskuje najbardziej trafne informacje.
- MAP@k (Mean Average Precision at k)

  - Ocenia średnią dokładność w kontekście wielu zapytań, co jest ważne w ewaluacji skuteczności odzyskiwania informacji.

Przykładowe modele embeddingowe

- **BERT (Bidirectional Encoder Representations from Transformers)** [10]:
  - BERT jest jednym z najbardziej znanych modeli do generowania embeddingów. Jest on szczególnie efektywny w rozumieniu kontekstu, co może być kluczowe w metodzie RAG.
- **GPT-2/GPT-3 (Generative Pretrained Transformer 2/3)** [11]:
  - Te modele, znane z generowania spójnego i kontekstowego tekstu, mogą być użyteczne do badania, jak generatywne embeddingi wpływają na proces generacji w RAG.
- **RoBERTa (A Robustly Optimized BERT Pretraining Approach)** [12]:
  - RoBERTa jest wariantem BERT, który został zoptymalizowany pod kątem większej dokładności. Może to zapewnić interesujące porównanie z tradycyjnym BERT-em.
- **DistilBERT (Distilled Version of BERT)** [13]:
  - Jest to uproszczona i bardziej efektywna wersja BERT-a pod względem obliczeniowym, co może być istotne w zastosowaniach, gdzie szybkość jest kluczowa.
- **T5 (Text-To-Text Transfer Transformer)** [14]:
  - T5 jest wszechstronnym modelem, który traktuje każde zadanie NLP jako zadanie konwersji tekstu na tekst, co może być interesujące w kontekście RAG.
- **ALBERT (A Lite BERT)** [15]:
  - ALBERT to uproszczona wersja BERT-a, która została opracowana przez Google w celu radzenia sobie z problemami wynikającymi z dużych rozmiarów modelu. Używa ona technik redukcji parametrów, co może być korzystne w kontekście efektywności obliczeniowej i skalowalności systemu RAG.

Wyżej wymienione przykładowe modele w większości zostały już uwzględnione w benchmarku MTEB. W związku z tym na platformie HuggingFace postaramy się znaleźć mniej znany model, który jeszcze nie został uwzględniony w benchmarku MTEB, a który naszym zdaniem ma szanse uzyskać sensowne wyniki.

Co ciekawe w leaderboardzie benchmarku MTEB w kategorii Retrieval wyróżnione są 3 podkategorie językowe - angielski, chiński oraz polski. W związku z tym być może zdecydujemy się skupić na podkategorii dedykowanej dla języka polskiego.

Wyniki eksperymentów

☐ TODO w dalszych etapach projektu

Wnioski

☐ TODO w dalszych etapach projektu

Źródła:

[1] Andrew Rosenberg and Julia Hirschberg. 2007. Vmeasure: A conditional entropy-based external cluster evaluation measure. pages 410–420.

[2] Muennighoff, Niklas, Nouamane Tazi, Loïc Magne, i Nils Reimers. "MTEB: Massive Text Embedding Benchmark." ArXiv:2210.07316.

[3] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models

[4] https://blog.langchain.dev/evaluating-rag-pipelines-with-ragas-langsmith/

[5] https://cobusgreyling.medium.com/rag-evaluation-9813a931b3d4

[6] https://www.linkedin.com/pulse/what-retrieval-augmented-generation-grow-right/

[7] Thulke, David, Nico Daheim, Christian Dugast and Hermann Ney. "Efficient Retrieval Augmented Generation from Unstructured Knowledge for Task-Oriented Dialog." ArXiv abs/2102.04643 (2021): n. pag.

[8] Lewis, Patrick, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler et al. "Retrieval-augmented generation for knowledge-intensive nlp tasks." Advances in Neural Information Processing Systems 33 (2020): 9459-9474.

[9] "BEIR-PL: Zero Shot Information Retrieval." ArXiv:2305.19840.

[10] Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." ArXiv:1810.04805.

[11] Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. "Language Models are Unsupervised Multitask Learners." OpenAI.

[12] Liu, Yinhan, Danqi Chen, et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach." ArXiv:1907.11692.

[13] Sanh, Victor, Lysandre Debut, Julien Chaumond, and Thomas Wolf. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter." ArXiv:1910.01108.

[14] Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer." Google, Mountain View, CA. ArXiv:1910.10683.

[15] Lan, Zhenzhong, Piyush Sharma, et al. “ALBERT: A Lite BERT for Self-supervised Learning of Language Representations.” Google AI. ArXiv:1909.11942.