Algorytmy dostępu do pamięci podręcznej dużych modeli językowych

Aliaksandr Marchuk

Politechnika Warszawska, Wydział Elektroniki i Technik Informacyjnych,
Nowowiejska 15/19, 00-665, Warszawa
01155186@pw.edu.pl

Streszczenie Duże modele językowe odgrywają kluczową rolę w nowoczesnej sztucznej inteligencji, umożliwiając naturalne i intuicyjne dialogi. Pomimo zaawansowanego zastosowania, napotykają one na różnorodne wyzwania, które mogą wpływać na ich efektywność. W odpowiedzi na te wyzwania rozwijane są różne metody mające na celu poprawę wydajności i dokładności modeli językowych. W niniejszej pracy chciałbym skupić się głównie na metodach powiązanych z różnymi algorytmami pamięci podręcznej dużego modelu językowego, szczególnie takich jak RAG [1] (Retrieval Augmented Generation) oraz grafach wiedzy. W celach praktycznego zrozumienia oraz aby ocenić proces działania, zalety i wady różnych algorytmów przeprowadzono eksperymenty stanowiące podzbiór tych które zostały zaproponowane w artykułach opisujących wybrane metody.
Słowa kluczowe: Duże modele językowe, RAG, Grafy wiedzy

1 Wprowadzenie

1.1 Wstęp

Duże modele językowe rewolucjonizują interakcje człowieka z systemami sztucznej inteligencji, umożliwiając bardziej naturalne i intuicyjne dialogi. Dzięki swojej zdolności do przetwarzania i generowania tekstu na podstawie ogromnych zbiorów danych, znajdują zastosowanie w wielu dziedzinach, takich jak wirtualni asystenci, edukacja oraz obsługa klienta. Pomimo ich dużego zastosowania, wciąż istnieje szereg wyzwań, który wpływa na ich efektywność i skalę zastosowania. Dlatego w swojej pracy chciałbym skupić się na przeanalizowaniu wybranych istniejących rozwiązań mających na celu częściowe zredukowanie niektórych z kluczowych problemów.

1.2 Opis rozdziałów

- **Rozdział 1 ("Wprowadzenie")**: wprowadza w temat artykułu oraz krótko przedstawia zawartość poszczególnych rozdziałów.
- **Rozdział 2 ("Wprowadzenie do dużych modeli językowych a typów pamięci podręcznej")**: omawia główne problemy związane z dużymi modelami językowymi oraz możliwe podejścia do ich rozwiązania.

- **Rozdział 3 ("Metodologia")**: prezentuje proces ewaluacyjny wraz z kryteriami sukcesu oraz analizą wybranych metod badawczych.
- **Rozdział 4 ("Opis platformy DeepView")**: przedstawia platformę DeepView, jej architekturę, opis funkcjonalności oraz sposobu wykorzystania.
- **Rozdział 5 ("Definicja i realizacja eksperymentów")**: szczegółowo opisuje realizację eksperymentów, przedstawiając wyniki i analizę.
- **Rozdział 6 ("Podsumowanie")**: podsumowuje najważniejsze wnioski z przeprowadzonych badań, omawiając osiągnięcia, wyzwania oraz możliwe kierunki dalszych badań.

2 Wprowadzenie do dużych modeli językowych a typów pamięci podręcznej

Na początku tej sekcji omówimy główne problemy związane z dużymi modelami językowymi, a następnie przedstawimy podejścia do ich rozwiązania.

1. **Halucynacje:** Modele mogą generować gramatycznie poprawne, ale merytorycznie błędne lub nieprawdziwe odpowiedzi.
2. **Generalizacja do nowych domen:** Modele mają trudności z adaptacją do nowych dziedzin wiedzy, które nie były objęte w danych treningowych.
3. **Wybór modelu:** Trening i uruchamianie niektórych dużych modeli językowych wymaga ogromnych zasobów obliczeniowych i pamięciowych, co powoduje wysokie koszty i ograniczenia w dostępności.
4. **Brak transparentności:** Modele działają jako "czarne skrzynki", co utrudnia interpretację wyników i diagnostykę błędów, co jest problematyczne w przypadkach kiedy chcemy zrozumieć dlaczego wyniki są dokładnie takie jak są a nie inne.

Ze względu na to że problem dotyczący **Braku transparentności** cześciowo należy do większej części zaawansowanych systemów szcztucznej inteligencji a problem dotyczący **Wyboru modelu** z czasem staje się mniej aktualny z powodu wzrostu dostępnych zasobów obliczeniowych czy też zwiększenia liczby dostawców takich rozwiązań, w swojej pracy chciałbym skupić się na praktycznych próbach rozwiązania pierwszych dwóch problemów.

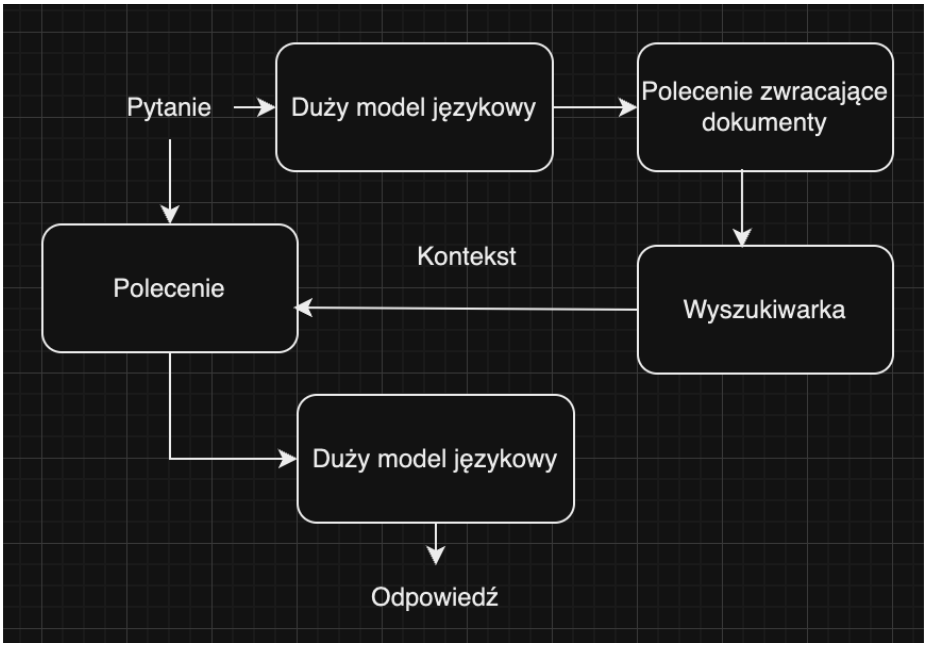
Istnieją różne skuteczne metody probujące poprawić dokładność generowanych odpowiedzi, niektóre z nich są opisane poniżej:

- **Algorytmy dostępu do pamięci podręcznej dużego modelu językowego:** Integracja mechanizmów weryfikujących generowane informacje w czasie rzeczywistym korzystając przy tym z zewnętrznych źródeł informacji, takich jak wektorowa baza danych czy grafy wiedzy.
- **Inżynieria promptów (Prompt Engineering):** Tworzenie i dostosowywanie poleceń, aby zapewnić kontekst i prowadzić model do generowania lepszych odpowiedzi.
- **Fine-tuning nadzorowany (Supervised Fine-Tuning):** Wykorzystanie ręcznie oznaczonych danych do poprawy dokładności dużego modelu językowego.

– **Systemy sprzężenia zwrotnego i wnioskowania (Feedback and Reasoning Systems):** Wykorzystanie iteracyjnego sprzężenia zwrotnego lub samooceny do poprawy początkowych odpowiedzi dużego modelu językowego.

Ze względu na to że różne rozwiązania wymagają implementacji różnego stopnia skomplikowaności architektur, w swojej pracy chciałbym skupić się na zadaniu inferecji, przy którym nie wymagany jest dodatkowy trening a zatem powinno to być prostsze w zaimplementowaniu, oraz szczególnie na dwóch algorytmach dostępu do pamięci podręcznej dużego modelu językowego.

2.1 Pamięć podręczna typu RAG



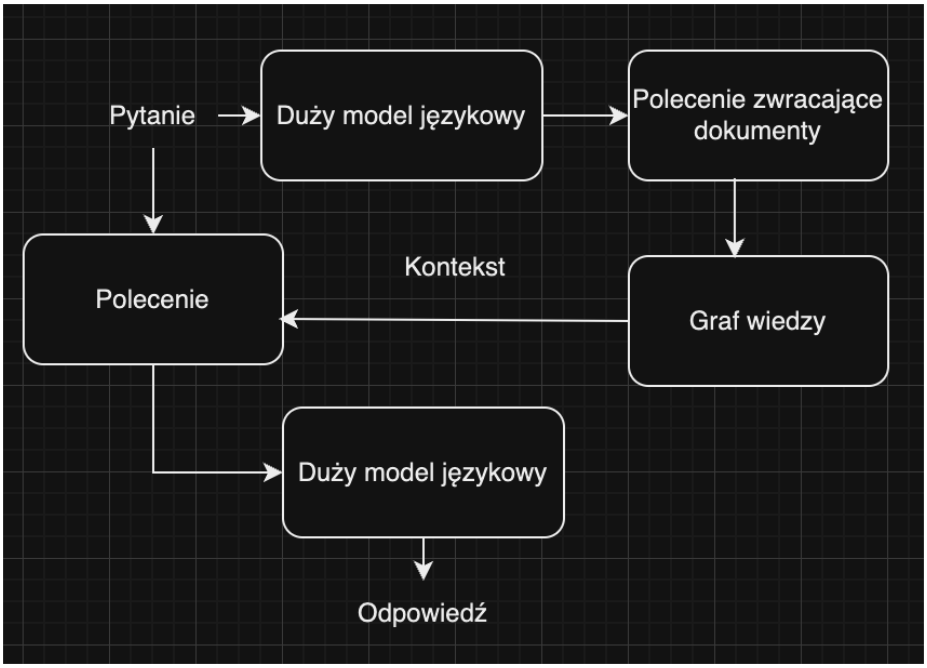
Rysunek 1. Diagram łańcucha RAG

Pamięć podręczna typu RAG integruje przeszukiwanie zewnętrznych źródeł informacji z generowaniem tekstu przez duże modele językowe. Proces ten obejmuje kilka kluczowych etapów:

1. **Tokenizacja:** Zapytanie użytkownika jest dzielone na mniejsze jednostki, nazywane tokenami, które następnie są konwertowane na reprezentacje numeryczne.

2. **Wyszukiwanie:** Tokenizowane zapytanie trafia do wyszukiwarki (Retriever), która przeszukuje zewnętrzne źródła informacji, identyfikując najbardziej odpowiednie dokumenty lub fragmenty tekstu na podstawie porównywania reprezentacji numerycznych zapytania a dokumentu.
3. **Łączenie:** Zwrócone dokumenty są łączone z oryginalnym zapytaniem w formie polecenia, które trafia do dużego modelu językowego.
4. **Generowanie odpowiedzi:** Duży model językowy wykorzystuje dostarczony kontekst do wygenerowania odpowiedzi, która jest bardziej merytorycznie trafna i spójna.

2.2 Pamięć podręczna grafowa



Rysunek 2. Diagram łańcucha z grafem wiedzy

Pamięć podręczna grafowa wykorzystuje grafy wiedzy jako zewnętrzne źródła informacji. Proces obejmuje następujące etapy:

1. **Tokenizacja:** Zapytanie użytkownika przekształcane są na reprezentację wektorową, co umożliwia dalsze przetwarzanie.
2. **Wyszukiwanie:** Zapytanie trafia do grafu wiedzy, aby znaleźć trójki (podmiot, predykat, obiekt) najbardziej związane z zapytaniem. Proces ten polega na porównaniu reprezentacji numerycznych zapytania i potencjalnych

trójek przy użyciu metryk podobieństwa, takich jak kosinusowa miara podobieństwa, aby zidentyfikować najbardziej odpowiednie informacje.
3. **Łączenie:** Zebrane trójki są dołączane do oryginalnego zapytania, tworząc bardziej rozbudowany kontekst.
4. **Generowanie odpowiedzi:** Duży model językowy wykorzystuje wzbogacony kontekst do wygenerowania odpowiedzi, która jest bardziej poprawna.

3 Metodologia

W sekcji tej najpierw przedstawiony jest ogólny zakres procesu ewaluacyjnego dla wybranej metody badawczej a później podane są odpowiednie szczegóły opisujące procesy ewaluacyjne wybranych metod.

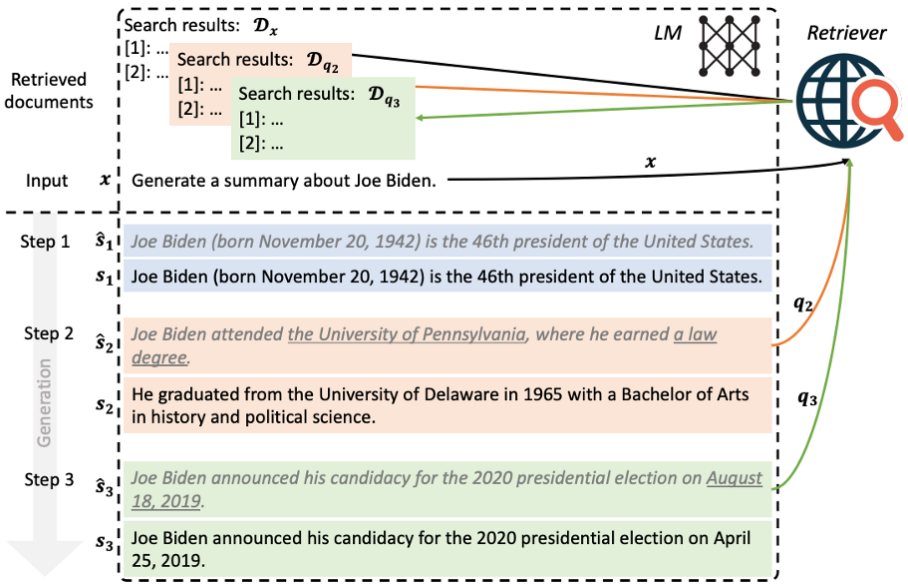
3.1 Proces ewaluacyjny:

1. **Model:** Wybór konkretnego modelu w zależności od potrzeb zadania oraz dostępnych zasobów obliczeniowych.
2. **Zbiory danych:** Wybór jednego, poszczególnych czy też dodatkowych zbiorów danych.
3. **Metryki:** Określenie metod oceny wyników, mogą to być takie same metryki jak w wybranej metodzie badawczej lub połączenie oryginalnych i dodatkowych ale oryginalne muszą pozostać.
4. **Model bazowy:** Wybór standardowego modelu jako punktu odniesienia dla późniejszej oceny kryteriów sukcesu.
5. **Kryteria sukcesu:** Definicja kryteriów sukcesu, które są miarą oceny poprawności wyników eksperymentów. Najcześciej można wystartować od tego że wyniki z zaimplementowaną metodą badawczą muszą być lepsze niż wyniki modelu bazowego, później zwiększać liczbę porównanych modeli.
6. **Eksperymenty oraz analiza wyników:** Przeprowadzenie eksperymentów w celu porównania wyników z uzyskanymi w artykułach przedstawiających wybrane metody badawcze.

3.2 Analiza wybranych metod badawczych

Opis metody FLARE:

Definicja: FLARE [2] (Forward-Looking Active Retrieval Augmented Generation) to metoda, która iteracyjnie wzbogaca generację tekstu poprzez aktywne wyszukiwanie i integrowanie zewnętrznych informacji, aby poprawić dokładność i spójność odpowiedzi. Metoda została stworzona przez Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan i Graham Neubig w 2023 roku.



Rysunek 3. Diagram działania metody FLARE

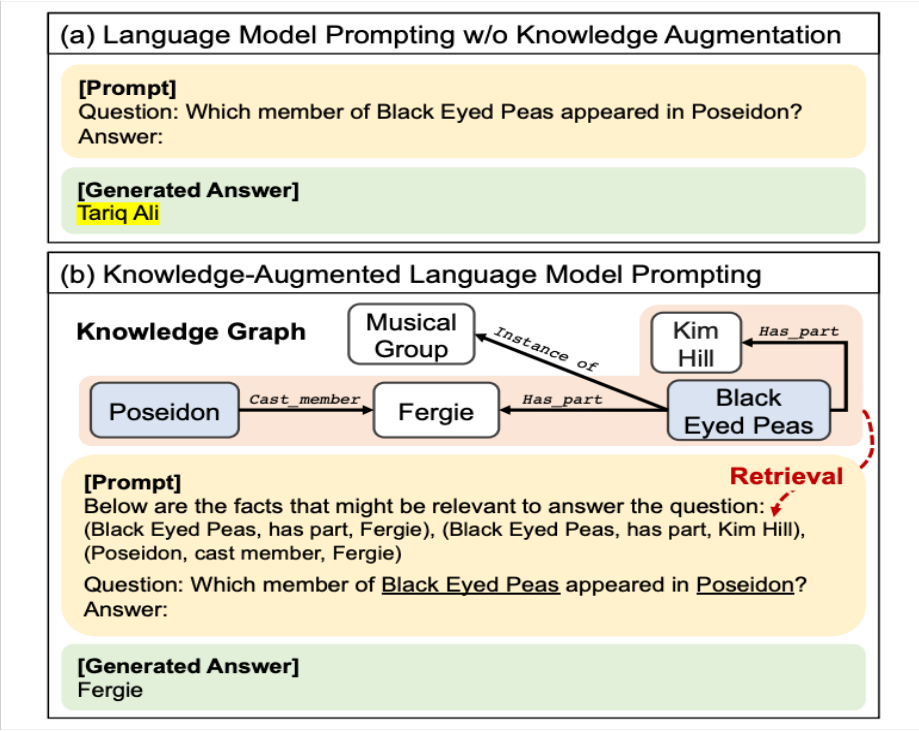
Sposób działania:

1. **Generowanie pierwszej propozycji:** Duży model językowy generuje wstępne zdanie na podstawie wejściowego zapytania, które jest sprawdzane pod kątem małoprawdopodobnych tokenów.
2. **Wyszukiwanie informacji:** Na podstawie zapytania i wstępnego zdania, wyszukiwarka przeszukuje zewnętrzne źródła informacji, zwracając najlepiej pasujące dokumenty.
3. **Regeneracja zdania:** Na podstawie wyszukanych dokumentów, duży model językowy generuje ostateczną wersję zdania.
4. **Iteracyjne generowanie:** Proces jest powtarzany dla kolejnych zdań, aż do wygenerowania kompletnej odpowiedzi. Każde zdanie jest generowane, sprawdzane i wzbogacane przy użyciu wyszukiwarki, aby zapewnić wyższą jakość i zgodność z zewnętrznymi źródłami informacji.

1. **Modele:**
   (a) text-davinci-003
2. **Typ pamięci podręcznej:**
   (a) RAG
3. **Komponenty:**
   (a) Wyszukiwarka: BM25 dla Wikipedii, Bing dla zadania sumaryzacji z dostępem do internetu.
   (b) Duży model językowy: text-davinci-003.

4. **Zbiory danych:**
   (a) 2WikiMultihopQA: Zestaw danych wielozadaniowych pytań wymagających wielokrotnego przeszukiwania Wikipedii.
   (b) StrategyQA: Zestaw danych pytań wymagających strategii wnioskowania.
   (c) ASQA: Zestaw danych pytań związanych z aspektami wiedzy.
   (d) WikiAsp: Zestaw danych dotyczących aspektów artykułów Wikipedii.
5. **Metryki:**
   (a) EM (Exact Match): Dokładne dopasowanie odpowiedzi.
   (b) F1: Harmoniczna średnia precyzji i recall.
   (c) ROUGE: Ocena jakości streszczeń tekstowych.
   (d) Disambig-F1: F1 z uwzględnieniem rozróżnienia między różnymi znaczeniami.
   (e) Named Entity-based F1: F1 oparty na poprawnym rozpoznaniu nazwanych jednostek.
   (f) UniEval: Uniwersalna ocena różnych aspektów odpowiedzi.

Opis metody KAPING:



Rysunek 4. Diagram działania metody KAPING

Definicja: KAPING [3] (Knowledge-Augmented Language Model PromptING) to metoda, która wzbogaca pytania wejściowe odpowiednimi faktami z grafu wiedzy, aby poprawić dokładność generowanych odpowiedzi przez modele językowe. Metoda została stworzona przez Jinheon Baek, Alham Fikri Aji i Amir Saffari w 2023 roku.

Sposób działania:

1. **Wyszukiwanie odpowiednich faktów:** System wyszukuje odpowiednie fakty z grafu wiedzy, porównując reprezentacje numeryczne zapytania a trójek wybranego grafu wiedzy.
2. **Integracja faktów z zapytaniem:** Znalezione fakty są łączone z pierwotnym zapytaniem.
3. **Generowanie odpowiedzi:** Wzbogacone zapytanie jest przetwarzane przez duży model językowy w celu wygenerowania bardziej precyzyjnej odpowiedzi.

1. **Modele:**
   (a) GPT-3
   (b) T5 (0.8B, 3B, 11B)
   (c) T0 (3B, 11B)
   (d) OPT (2.7B, 6.7B)
   (e) AlexaTM (20B)
2. **Typ pamięci podręcznej:**
   (a) Grafowa
3. **Komponenty:**
   (a) Wyszukiwarka: wyszukiwanie odpowiednich faktów z grafu wiedzy na podstawie podobieństw semantycznych.
   (b) Duży model językowy: modele językowe takie jak GPT-3, które wykorzystują wzbogacone zapytania do generowania odpowiedzi.
4. **Zbiory danych:**
   (a) WebQuestionsSP (WebQSP): Zestaw danych zaprojektowany z Freebase KG na podstawie Wikidata.
   (b) Mintaka: Zestaw danych zaprojektowany z Wikidata KG do złożonych zadań KGQA, zawierający 4,000 próbek w języku angielskim.
5. **Metryki:**
   (a) **Dokładność:** Procent poprawnie sklasyfikowanych przypadków w stosunku do wszystkich przypadków.
   (b) **Precyzja:** Stosunek liczby prawdziwie pozytywnych wyników do sumy prawdziwie pozytywnych i fałszywie pozytywnych wyników.
   (c) **Czułość:** Stosunek liczby prawdziwie pozytywnych wyników do sumy prawdziwie pozytywnych i fałszywie negatywnych wyników.
   (d) **Miara F1:** Harmoniczna średnia precyzji i czułości, zapewniająca zrównoważoną ocenę modelu.

4 Opis platformy DeepView

1. **Struktura projektu:**

```
DEEPVIEW/
|-- data/
|   |-- *.json
|-- results/
|   |-- *.json
|-- wikiasp/
|-- wikidata_retriever.py
|-- bing_retriever.py
|-- my_flare_chain.py
|-- my_kaping_chain.py
|-- README.md
|-- requirements.txt
|-- .gitignore
|-- LICENSE
```

   - **wikidata_retriever.py:** Skrypt odpowiedzialny za wyszukiwanie informacji z Wikidata. Implementuje metody do pobierania i przetwarzania danych z grafu wiedzy.
   - **bing_retriever.py:** Skrypt do wyszukiwania informacji z Bing. Zawiera funkcje do integracji wyników wyszukiwania z Bing z modelami językowymi.
   - **my_flare_chain.py:** Implementacja łańcucha FLARE. Zawiera kod do przeprowadzenia eksperymentów oraz obsługę zbiorów danych.
   - **my_kaping_chain.py:** Implementacja łańcucha KAPING. Zawiera kod do przeprowadzenia eksperymentów oraz obsługę zbiorów danych.
   - **README.md:** Plik readme zawierający podstawowe informacje o projekcie, instrukcje dotyczące instalacji oraz uruchamiania skryptów.
   - **requirements.txt:** Plik zawierający listę zależności projektu. Określa wymagane biblioteki Python, które muszą zostać zainstalowane.
   - **.gitignore:** Plik konfiguracyjny dla Git, określający które pliki i foldery powinny być ignorowane przez system kontroli wersji.
   - **LICENSE:** Plik zawierający licencję projektu.

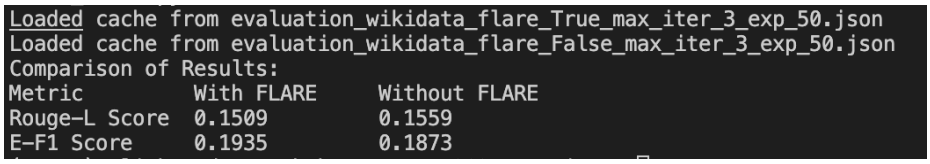
2. **Cel projektu:** Stworzenie uniwiersalnej platformy eksperymentalnej oferującej łatwe modyfikacji aby uzyskać pythonową implementację odpowiadającej w procesie działania dowolnej metodzie badawczej, kluczowe elementy której to: **duży model językowy** (MyFLAREChain i MyKAPINGChain, i też inne, będą dziedziczyły z klasy szablonowej w przyszłości), **wyszukiwarka** (BingRetriever czy WikidataRetriever) oraz **zbiory danych** do których będą dorobione oddzielne klasy (na razie kod jest wewnątrz odpowiednich plików z kodem źródłowym łańcuchów).

5 Definicja i realizacja eksperymentów

Pliki źródłowe dostępne są na GitHub https://github.com/Mamba369/DeepView. Eksperymenty przeprowadzono za pomocą modelu **gpt-3.5-turbo-instruct** od firmy **OpenAI** [4].

5.1 Proces ewaluacyjny oraz wyniki eksperymentów metody FLARE

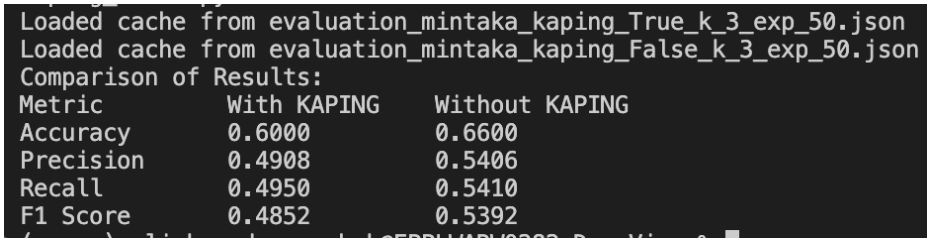
1. **Model:** gpt-3.5-turbo-instruct
2. **Zbiory danych:** WikiASP
3. **Metryki:** E-F1 i Rouge-L



Rysunek 5. Wyniki eksperymentów metody FLARE dla 50 próbek WikiASP

5.2 Proces ewaluacyjny oraz wyniki eksperymentów metody KAPING

1. **Model:** gpt-3.5-turbo-instruct
2. **Zbiory danych:** Mintaka
3. **Metryki:** Dokładność, precyzja, czułość, miara F1



Rysunek 6. Wyniki eksperymentów metody KAPING dla 50 próbek Mintaka

Natomiast w przypadku procesu ewaluacyjnego dla metody KAPING dla małej liczby przypadków (15 próbek) duży model językowy odpowiadał lepiej

z zaimplementowaną metodą niż bez, co oznacza że generowane przez wyszukiwarkę trójki rzeczywiście poprawiły jakość działania modelu.

```
Loaded cache from evaluation_mintaka_kaping_True_k_3_exp_15.json
Loaded cache from evaluation_mintaka_kaping_False_k_3_exp_15.json
Comparison of Results:
Metric         With KAPING    Without KAPING
Accuracy       0.7333         0.6667
Precision      0.7885         0.6607
Recall         0.7885         0.6607
F1 Score       0.7538         0.6286
```

Rysunek 7. Wyniki eksperymentów metody KAPING dla 15 próbek Mintaka

6 Podsumowanie

Pierwsze rezultaty wymagają wykonania dodatkowych badań i testów oraz służą jako dobry punkt startowy dla przyszłych modyfikacji poprawiających cały proces działania oraz wyniki.

Literatura

1. Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, Douwe Kiela, *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. 2020. https://arxiv.org/abs/2005.11401
2. Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, Graham Neubig, *FLARE: Forward-Looking Active Retrieval Augmented Generation*. 2023. https://arxiv.org/abs/2305.06983
3. Jinheon Baek, Alham Fikri Aji, Amir Saffari, *KAPING: Knowledge-Augmented Language Model PromptING*. 2023. https://arxiv.org/abs/2306.04136
4. OpenAI, *Models Documentation*. Available at: https://platform.openai.com/docs/models