Data Mining Project

February 18, 2024

1 Echipa

- Tema aleasă: Proiectul default
- Specializare: Baze de date
- Membrii echipei:
  - Apetrei Roxana
  - Barabas Elina
  - Bordei Amalia
  - Danicico George
  - Suciu Andrei

2 Descrierea codului

Codul din proiect implementează un sistem de indexare și căutare a conținutului din fișierele Wikipedia, utilizând o combinație de prelucrare a textului, analiză de limbaj natural și tehnologii de indexare de tip motor de căutare.

Codul este alcătuit din mai multe clase:

- **Index** - Aceasta clasa se ocupa cu crearea si gestionarea indexului.

  In momentul in care se creeaza o instanta pentru aceasta clasa, se creeaza si indexul oferit de biblioteca Whoosh. Daca exista folderul 'my_index_directory' in folderul proiectului, indexul se creeaza pe baza acestuia, in caz contrar se creeaza pe baza fisierelor Wikipedia din folderul 'data/FileExample'.

  In functia 'process_file(filename)' se proceseaza cate un fisier Wikipedia. Se creeaza un dictionar in care se stocheaza pentru fiecare titlu, continutul documentului respectiv. Pe langa impartirea in titlu si document se aplica anumite preprocesari prezentate in capitolul:

```
            if line.startswith("[[") and not
                line.startswith(Constants.IMAGE_TAG) \
                and not line.startswith(Constants.FILE_TAG) \
                        and line.endswith("]]\n"):
                if current_title:
                    content[current_title] = body
                    body = ""
                    current_title = line[2:-3]
                elif current_title and line.strip():
                    processed_line = self.__process_line(line.strip())
                    body += " " + processed_line

    def __process_line(line):
        if line.startswith("==") and line.endswith("=="):
            num_eq = line.lstrip('=').count('=')
            processed_line = line[num_eq:-num_eq]
        elif line.startswith(Constants.REDIRECT_PREFIX):
            processed_line = line[10:]
        elif Constants.HTTP_PREFIX in line:
            processed_line = line.replace(Constants.HTTP_PREFIX, "")
        elif Constants.HTTPS_PREFIX in line:
            processed_line = line.replace(Constants.HTTPS_PREFIX, "")
        else:
            processed_line = line

        return processed_line
```

Pentru indexul nostru vom folosi ca si analizator StemmingAnalyzer, oferit de Whoosh. In cadrul acestuia se face tokenizarea inputului in functie de regex-ul dat, se transforma token-ele in litere mici, se aplica stemming pe fiecare token si la final se elimina StopWords-urile. StopWords-urile au fost importate din biblioteca 'nltk'.

```
my_analyzer = StemmingAnalyzer(expression=r"[\w-]+(\.?\w+)*",
stoplist=self.__stop_words)
```

Pentru a creea schema indexului nostru vom folosi clasa Schema din Whoosh. Schemă specifică două câmpuri:

- title: Un câmp pentru titlurile documentelor. Este configurat să fie indexat și stocat, iar textul este procesat folosind analizatorul my_analyzer. De asemenea, acest câmp are un boost de 2.0, ceea ce înseamnă că va avea o importanță mai mare în căutările ulterioare.
- content: Un câmp pentru conținutul documentelor. Este configurat să fie doar indexat (nu și stocat), iar textul este procesat folosind același analizator my_analyzer.

```
schema = Schema(title=TEXT(stored=True, analyzer=my_analyzer,
field_boost=2.0), content=TEXT(analyzer=my_analyzer))
```

O alta functie importanta este functia 'retrive(query)'. Aceasta primeste ca si parametru un query, de tip Query din biblioteca Whoosh si returneaza lista de titluri de pagini Wikipedia sortate descrescator in functie de scor.

- **IntexTest** - In aceasta clasa se testeaza si se masoara rezultatele pentru indexul nostru. Toate acestea se intampla in functia 'test_index()'. Initial se incarca intrebarile din fisierul 'questions.txt'. Pentru fiecare query vom folsi QueryParserul oferit de catre Whoosh:

  ```
  QueryParser("content", schema=self.__myIndex.get_index().schema,
  group=OrGroup)
  ```

  Dupa se calculeaza metricile prezentate in capitolul 3.

- **Constants** - In aceasta clasa avem salvate toate constantele din proiect.

- **ChatGptUtil** - In aceasta clasa avem intreactiunea proiectului cu ChatGPT. Avem o singura functie, 'chat_with_gpt(user_content)' care iterogheaza ChatGPT-ul.

In functia 'main' din fisierul 'main.py' se instantiaza obiectele si se ruleaza functia 'test_index' din clasa TestIndex.

În linii mari, acest cod îmbină procesarea textului, indexarea eficientă și interacțiunea cu modele de limbaj avansate pentru a crea o soluție comprehensivă de căutare și răspuns la întrebări într-un volum mare de date.

3 Cerințele proiectului

3.1 Indexing and retrieval

3.1.1 Pregatirea termenilor pentru indexare

Înainte de a crea indexul, am adoptat o abordare de preprocesare care a implicat extragerea atentă a informațiilor relevante din fiecare fișier, concentrându-ne pe titlu și conținut. Această selecție meticuloasă a datelor a fost gândită pentru a simplifica și optimiza procesul de indexare ulterior.

In cadrul proiectului am urmat urmatorii pasi de procesare a termenilor inainte de indexare:

- Am tokenizat textul si am eleminat caracterele speciale, cele ce difera de litere si cifre.

- Am transformat tokenii in litere mici.

- Eliminam StopWords-urile.

- Am aplicat stemming pe fiecare token.

Toate acestea au fost facute cu ajutorul clasei StemmingAnalyzer din biblioteca Whoosh.

3.1.2 Probleme întâlnite

Înainte de creearea index-ului, am întâmpinat mai multe probleme cu conținutul fișierelor de wikipedia.

- Structura inconsistentă a fișierelor.
- Fișiere care conțineau linii care încep cu prefixul *#REDIRECT* .
  Ex: #REDIRECT Balfour Declaration [tpl]R from other capitalisation[/tpl]
- Prezenta diferitelor link-uri care au prefixul *http://www* sau *https://www*.
  Ex: http://www.kitemark.com/about-kitemark/
- Prezenta tag-urilor *[tpl]*.
  Ex: [tpl]R from other capitalisation[/tpl]
- Prezenta tag-urilor *[ref]*
  Ex:[ref]Peirce Jastrow (1885)[/ref]
- Prezenta tag-urilor *[[Image:*
  Ex:[[Image:Cameroon boundary changes.PNG—thumb—right—Cameroon over time]]
- Prezenta tag-urilor *[[File:*
  Ex:[[File:Croatia economy chart.png—thumb—left—230px—Croatian development in the 20th century]]
- Fișiere care conțineau caractere speciale (mai ales cele în chineză și germană).

În timpul procesului de creare a indexului, am întâmpinat dificultăți semnificative legate de performanță, întrucât timpul necesar pentru finalizarea acestei operațiuni a fost considerabil îndelungat. Această întârziere a fost accentuată de limitările resurselor hardware ale laptopurilor noastre.

3.2 Descrieți cum ați construit queryul

Pentru a completa căutarea, am combinat întrebarea cu categoria prin concatenarea celor două string-uri și apoi am aplicat aceeași preprocesare utilizată pentru documentele Wikipedia. În plus față de aceasta, am inclus și un 'OrGroup' în query-ul nostru, permițând căutarea documentelor care conțin oricare dintre tokenii specificați, fără a fi necesară prezența tuturor tokenilor din query. Aceasta asigură o căutare mai flexibilă și mai robustă, cu posibilitatea de a returna rezultate relevante.

3.3 Măsurarea performanței modelului

Pentru a putea masura performanta modelului am ales urmatoarele metrici:

- **Precision at one(P@1)**:
  Am ales această metrică pentru evaluarea performanței în cadrul sistemului Jeopardy deoarece oferă o măsură clară și relevantă a capacității sistemului de a furniza răspunsuri precise și relevante într-un timp scurt. Într-un sistem de tip Jeopardy, utilizatorii așteaptă un singur răspuns, considerat cel mai adecvat sau corect pentru întrebarea formulată.

  $$\text{Precision at One} = \frac{\text{Numărul de întrebări pentru care răspunsul corect este primul}}{\text{Numărul total de întrebări}} \times 100$$

  Pentru aceasta metrica, sistemul nostru a obtinut valoarea: 0.2.

- **Mean Reciprocal Rank (MRR)**: 0.259
  Am ales aceasta metrica pentru sistemul nostru doarece MRR evaluează performanța sistemului prin măsurarea poziției primului răspuns corect în lista de răspunsuri returnată. Chiar daca sistemul nu returneaza raspunsul corect pe prima pozitie, ne intereseaza ca raspunsul corect sa fie cat mai sus in lista returnata.

  $$\text{MRR}(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{\text{rank}_j}$$

  Pentru aceasta metrica, sistemul nostru a obtinut valoarea: 0.267

Aceste rezultate furnizează o imagine a performanței indexului în gestionarea întrebărilor și oferă puncte de referință pentru a ghida optimizările ulterioare. De exemplu, analiza diferențelor dintre precizia obișnuită și cea cu ChatGPT poate sugera modul în care interacțiunea cu modele de limbaj avansate poate îmbunătăți experiența de căutare și răspuns.

3.4 Analiza erorilor

3.4.1 Cate intrebari au fost raspunse corect/incorect?

Index-ul pe care l-am creat a furnizat răspunsuri corecte la 20 din cele 100 de întrebări pe care le-a primit.

3.4.2 De ce poate un model atât de simplu să răspundă corect la unele intrebări?

Într-un context în care întrebările sunt formulate în mod clar și au o structură simplă, modelele mai puțin complexe pot răspunde corect. Acestea pot beneficia de o structură simplă a întrebării, care facilitează înțelegerea și furnizarea unui răspuns adecvat. De asemenea, dacă întrebările au o relevanță directă și se

referă la informații faptuale sau cunoscute, modelele simple pot accesa aceste informații și returna răspunsuri precise.

În plus, un set de date bine structurat poate contribui la succesul unui model simplu. Dacă modelul este antrenat pe date curate și bine organizate, reflectând tipurile de întrebări cu care este confruntat, acesta poate oferi răspunsuri mai precise și coerente.

Cu toate acestea, limitele modelului simplu devin evidente în situații care necesită înțelegere complexă a contextului sau abilități de analiză mai profunde.

3.4.3 Ce probleme ati abservat la intrebarile raspunse incorect?

Analizand raspunsurile incorecte, am observat ca in cea mai mare parte cea mai mare problema este formularea intrebarii. Astfel, intrebarea difera foarte mult de continutul paginii Wikipedia corespunzatoare.
Aceasta problema apare in special la intrebarile in care sunt date citate. Spre exemplu, la intrebarea: 'Song that says, "you make me smile with my heart; your looks are laughable, unphotographab"', raspunsul corect este 'My Funny Valentine'. Dar analizand continutul paginii 'My Funny Valentine' observam ca tokenii difera in totalitate fata de in intrebare.
O alta problema semnalizata este categoria intrebarii. Spre exemplu, cea mai problematica categorie este 'STATE OF THE ART MUSEUM (Alex: We'll give you the museum. You give us the state.)'. Astfel, sistemul nostru va raspunde, in special, cu nume de fisiere Wikipedia ce contin muzee, in loc de state.
Pe langa categoria de mai sus, mai avem cateva categorii problematice: "CAPITAL CITY CHURCHES (Alex: We'll give you the church. You tell us the capital city in which it is located.)" si "COMPLETE DOM-INATION(Alex: Not "domination.")".

3.5 Imbunatatirea raspunsurilor

Folosind indexul nostru observam ca avem o precizie destul de mica(0.2). Astfel doar 20 de intrebari au primit raspuns corect.
Analizand si valoarea MRR-ului(0.26) putem concluziona ca raspunsurile corecte se pot afla pe pozitii apropiate de prima. Astfel, pentru a mari precizia ne vom folosi de ChatGPT. Selectam primele K=10 documente, returnate de index, si le transmitem la ChatGPT, impreuna cu intrebarea. Deoarece avem un numar limitat de tokeni in request-ul catre ChatGPT, vom folosi doar titlurile celor 10 documente returnate.
Astfel ii vom transmite lui ChatGPT sa alege doar una dintre variantele transmise de noi, nu alta varianta.
Folosind acest aproach am reusit sa crestem considerabil precizia sistemului, obtinand valoarea 0.55 pentru Precission at one.

4 Concluzii

Codul implementat a combinat prelucrarea textului, analiza de limbaj natural și tehnologii de indexare pentru a oferi o soluție comprehensivă de căutare și răspuns la întrebări. Acesta a parcurs fiecare fișier Wikipedia, extrăgând informațiile relevante pentru a crea un index eficient și precis.

În procesul de pregătire a datelor pentru indexare, am întâmpinat provocări legate de structura inconsistentă a fișierelor, prezența unor elemente nedorite precum redirect-uri și link-uri, precum și gestionarea caracterelor speciale.

Rezultatele obținute au furnizat metrici semnificative, care au reflectat performanța indexului nostru în furnizarea răspunsurilor corecte. Introducerea interacțiunii cu modelul ChatGPT a adus îmbunătățiri semnificative în precizia răspunsurilor.

Pe parcursul proiectului, am constatat că un model simplu poate furniza răspunsuri corecte în cazul întrebărilor clare și structurate. Cu toate acestea, limitele acestui model devin evidente în fața complexității întrebărilor și a necesității unei înțelegeri mai profunde a contextului.

În concluzie, proiectul nostru a reprezentat o provocare amplă și captivantă în domeniul Data Mining, evidențiind importanța preprocesării datelor, gestionării problemelor specifice ale conținutului Wikipedia și explorării sinergiilor dintre modele simple și avansate. Aceste experiențe ne-au consolidat înțelegerea proceselor de indexare și căutare și ne-au oferit o bază solidă pentru a aborda provocări viitoare în acest domeniu în continuă dezvoltare.