Proiect Data Mining

22.01.2024

Detalii despre echipă

- Tema aleasă: Proiectul Default
- Specializare master: Sisteme Distribuite în Internet, grupa 244
- Membrii echipei:
  - Deac Laura-Alexandra
  - Gal Paula-Maria
  - Doroftei Victor
  - Farcașanu Ștefan-Lucian
  - Giosan George

1 Descrierea codului

Codul este împărțit în mai multe clase, acestea fiind:

- **Indexer** - clasa ce se ocupă cu crearea indexului;
- **IndexViewer** - clasa responsabilă cu vizualizarea indexului și a conținutului documentelor;
- **Search** - clasa ce se ocupă cu performarea query-urilor pe index;
- **Lemmatizer** - clasa utilizată pentru lematizarea indiciilor și a conținutului documentelor;

- **JeopardyClueSearch** - clasa ce are ca scop iterarea prin întrebările de Jeopardy și rularea query-urilor bazate pe acestea.

Vom prezenta secvențele de cod utilizate pentru a răspunde la întrebările din cerințele proiectului (secțiunile **3.1** și **3.2**).

Pentru întrebarea **3.1.1**, am utilizat metoda `normalize(input)` din clasa `Indexer`, scopul acesteia fiind de a aplica pașii de preprocesare pe text.

În legătură cu problemele datelor de pe Wikipedia (întrebarea **3.1.2**), am utilizat câțiva RegEx pentru a filtra aceste secvențe nedorite de text, în metoda `main()` a clasei `Indexer`.

```
// Perform RegEx operations on the contents of the
   files (more details in the documentation).

content = content.replaceAll("\\[\\[File:.*]]]]",
   "");
content = content.replaceAll("\\[\\[File:.*]]", "");

content = content.replaceAll("\\[\\[Image:.*]]]]",
   "");
content = content.replaceAll("\\[\\[Image:.*]]",
   "");

content = content.replaceAll("\\[ref].*\\[/ref]",
   "");
content = content.replaceAll("\\[tpl].*\\[/tpl]",
   "");
```

Construirea query-ului (secțiunea **3.1.3**) a fost realizată în metoda `searchAllClues()` din cadrul clasei `JeopardyClueSearch`, mai exact:

```
Integer rank = search.getRankForClue(category + " "
   + clue, answer, titles);
```

Valoarea folosita drept query este normalizată în metoda `getResultsForQuery(clue)` a clasei `Search`:

```
clue = Indexer.normalize(clue);
```

Pentru măsurarea performanței, am realizat un calcul al metricii **Precision at one(P@1)**, realizată în metoda `searchAllClues()` a clasei `JeopardyClueSearch`:

```
Integer rank = search.getRankForClue(category + " "
   + clue, answer, titles);

if (rank == 1) {
    pAtOne++;
}

...

pAtOne /= numQuestions;
```

2 Rezultatele obținute

Rezultatele obținute de către modelul nostru se pot observa în documentul **jepardy_initial_output.txt**.

3 Întrebările din cerința proiectului

3.1 Cerința 1)

3.1.1 Descrieți cum ați pregătit termenii pentru indexare

Pentru a ne asigura că indexul dezvoltat o să fie unul bun și eficient, am fost nevoiți să supunem documentele mai multor pași de preprocesare. Astfel, pașii din preprocesare sunt:

- Tot textul din document a fost transformat să fie scris cu litere mici. Aici am folosit:

  ```
  public String toLowerCase()
  ```

- Am tokenizat textul folosind StandardTokenizer din Apache Lucene `https://lucene.apache.org/core/6_6_0/core/org/apache/lucene/analysis/standard/StandardTokenizer.html`
- Eliminăm stop words folosind un StopFilter din Apache Lucene `https://lucene.apache.org/core/8_0_0/analyzers-common/org/apache/lucene/analysis/core/StopFilter.html`. Lista de stop words pe care o folosim este:

```
"a", "an", "and", "are", "as", "at",
   "be",
"but", "by", "for", "if", "in", "into",
   "is",
"it", "no", "not", "of", "on", "or",
   "such",
"that", "the", "their", "then", "there",
"these", "they", "this", "to", "was",
   "will",
"with"
```

- Lematizăm textul folosind librăria StandfordCoreNLP `https://stanfordnlp.github.io/CoreNLP/lemma.html`.

După ce un document este preprocesat, acesta este trimis mai departe către index.

3.1.2 Ce probleme ați găsit cu datele de pe Wikipedia și cum le-ați rezolvat?

Din păcate ne-am lovit de mai multe probleme legate de conținutul documentelor în momentul în care încercam să pregătim indexul. Printre problemele menționate anterior se numără:

- Prezența structurilor / tagurilor de forma: **[ref]...[/ref]**

  Ex: [ref]For a discussion of Newton's original argument, see [/ref]

- Prezența structurilor / tagurilor de forma: **[tpl]...[/tpl]**

  Ex: [tpl]Clarify—date=March 2013[/tpl]

- Prezența structurilor / tagurilor de forma: **[[File: ...]]**

  Ex: [[File:Eastern Frontier, Cape of Good Hope, ca 1835.png—thumb—Eastern frontier of the colony, c. 1835]]

- Prezența structurilor / tagurilor de forma: **[Image:]**

  [[Image:World Map Index of perception of corruption.svg—thumb—300px——Overview of the index of perception of corruption, 2013.]]

Pentru a încerca să rezolvăm problemele menționate anterior, am fost nevoiți să dezvoltăm mai multe regexuri pentru a elimina din fișier aceste elemente ce nu ne ajutau. Totodată, nu am putut să eliminăm complet aceste probleme pentru că am descoperit mai multe inconsistențe în documentele primite ca și input, precum:

```
===Human Development Index===

[[File:2013 UN Human Development Report Quartiles.svg|300px|thumb|World map by quartiles of Human Development Index in 2013.
Countries in descending order of Human Development Index (2013 data):

===Globalisation===

The index of globalization in Central European countries (2013 data)[tpl]dead link|date=July 2013[/tpl]

===Prosperity Index===

Legatum Prosperity Index demonstrates an average and high level of prosperity in Central Europe:http://www.prosperity.com/
```

În acest caz, se poate observa că tagul **File** nu se închide, lucru ce ne încurcă în momentul in care dorim să îl eliminăm, pentru că nu se găsește un match (lipsesc caracterele "]]" la final).

3.1.3 Descrieți cum ați construit queryul

Pentru a forma queryul, am decis să ne folosim atât de indiciul inițial, cât și de categoria acestuia. După ce am concatenat cele 2 stringuri, am fost nevoiți să preprocesăm noul string obținut la pasul anterior. Rezultatul obținut la pasul anterior este de fapt queryul pe care o să îl folosească indexul.

3.2 Cerința 2)

3.2.1 Măsurarea performanței modelului

O metrică care ni s-a părut relevantă pentru evaluarea performanței proiectului este **Precision at one(P@1)**, care este deobicei utilizată în contextul în care este important să identificăm un singur răspuns corect dintr-un set de opțiuni (practic ce se cere în cadrul concursului Jeopardy). Valoarea obținută pentru întrebările din document este de **0.23**.

3.3 Cerința 3)

3.3.1 Câte întrebări au fost răspunse corect/incorect?

Din totalul celor 100 de întrebări, programul nostru a reușit să răspundă corect la 23 din ele.

3.3.2 De ce un sistem atât de simplu poate oferi răspunsurile corecte pentru unele întrebări?

Noi credem că un astfel de sistem răspunde bine la unele întrebări din cauza faptului că unele întrebări sunt formulate într-un mod foarte concis, unele dând impresia că sunt luate cuvânt cu cuvânt de pe Wikipedia. Un exemplu de astfel de întrebare ar fi: *Several bridges, including El Tahrir, cross the Nile in this capital* - care are răspunsul *Cairo*.

3.3.3 Ce probleme s-au observat pentru întrebările care au fost răspunse incorect?

Am încercat să analizăm motivele pentru care programul răspunde greșit la unele întrebări. Unele întrebări sunt formulate diferit față de altele, în sensul că cele cu “” contin anumite citări/fragmente/versuri, iar aceste “” sunt eliminate în partea de lematizare a conținutului documentului.

Încă un lucru pe care l-am observat este că anumite pagini de pe Wikpedia conțin informații asemănătoare, de exemplu pentru indiciul următor *“The High Kirk of St. Giles, where John Knox was minister”* răspunsul este *“Edinburgh”*, dar sistemul nostru pune documente cum ar fi *“John Knox”*, *“Scotland”*, *“History of Scotland”* sau *“Glasgow”* pe poziții mai bune decât răspunsul corect.

Nu în ultimul rând, deoarece considerăm ca și răspuns pentru o întrebare titlul primei pagini returnate de către sistemul nostru, am identificat întrebări a căror răspunsuri nu reprezintă titlul unei pagini de Wikipedia sau faptul că acea pagină nu există printre paginile indexate de către noi.

Acum că am descris toate problemele identificate de noi, o grupare a acestora în clase ar fi:

- Probleme la formularea indiciului:
  - Indicii ce conțin diferite caractere speciale (””, ?, !, ...).
  - Indicii ce sunt formulate vag.

- Probleme la partea de documente:
  - Documente similare.
- Probleme la răspunsuri
  - Răspunsuri care nu se găsesc ca și titluri pentru o pagină de Wikipedia.
  - Răspunsuri a căror pagină de Wikipedia nu a fost indexată.

3.4 Cerința 4)

După efectuarea testelor inițiale, am observat că pentru o parte semnificativă din întrebări (20 de întrebări), răspunsul corect nu se regăsește pe rankul 1, ci în primele câteva poziții (pozițiile 2-5). Acest aspect ne-a dus cu gândul de a calcula valoarea metricii *MRR*. *MRR* este o metrică de performanță des folosită în sistemele de IR pentru a evalua eficiența algoritmului de căutare și de ierarhizare a documentelor. *MRR* ține cont de ordinea în care sunt returnate rezultatele corecte. Este important ca informația relevantă să fie prezentată cât mai sus în lista de rezultate.

În cadrul programului nostru, valoarea metricii **MRR** este egală cu **0.313**, ceea ce înseamnă că în medie, răspunsul corect pentru o întrebare se află aproximativ în primele 3 documente ierarhizate. Astfel, putem utiliza următoarea idee: vom folosi sistemul nostru pentru a obține primele K cele mai relevante documente pentru fiecare întrebare, iar pe acestea le vom re-ierarhiza, utilizând ChatGPT, în speranța că răspunsul corect va fi pus pe prima poziție de către acesta.

Astfel, pentru fiecare întrebare am selectat primele K=5 documente, pentru că am observat că primele 43 de documente aveau răspunsul corect ierarhizat în primele 5 documente, dar și datorită limitărilor ChatGPT, care nu poate lua ca și input un număr nelimitat de fișiere. După ce am efectuat acest proces, am observat că ChatGPT păstrează răspunsul corect pentru întrebările răspunse corect de către programul nostru, iar pentru aproape toate întrebările răspunse greșit, acesta pune răspunsul corect pe prima poziție în noua ierarhie (a greșit răspunsul pentru 2 întrebări).

Astfel, valoarea metricii **P@1** este îmbunătățită la valoarea **0.41**, ceea ce înseamnă că performanța sistemului s-a îmbunătățit cu aproximativ **78%**.

Prompturile pe care le-am folosit pentru a îi comunica lui ChatGPT să ne re-rankeze primele 5 documente pentru fiecare întrebare se găsesc la linkurile

următoare: https://chat.openai.com/share/d753bb40-ada0-495b-8883-37de050ce346 (întrebările 0-22), https://chat.openai.com/share/170569f5-4478-4a6d-a4b7-63d0e54c78d9 (întrebările 23-42). În cazul în care nu merge linkul anterior, în cadrul repoului de pe GitHub în folderul **chatgpt**, pot fi găsite paginile HTML aferente.