Proiect 1
Indexare şi căutare

an univ. 2018 – 2019

1 Prezentarea temei de proiect

Prima componentă de proiect a disciplinei „*Regăsirea Informaţiilor pe WEB*” vă propune să realizaţi un set de modificări asupra aplicaţiilor dezvoltate pe parcursul laboratoarelor 1 – 4 astfel încât:

1. să obţineţi o procesare adecvată a cuvintelor determinate în cadrul unui text;
2. să studiaţi şi alte tipuri de baze de date şi mecanisme de stocare de date (precum MongoDB);
3. să studiaţi alte metode de realizare a operaţiilor de căutare.

Astfel, pentru prima cerinţă de mai sus, metodele de construire a indecşilor direcţi şi inverşi trebuie să implementeze mecanisme prin intermediul cărora un cuvânt de dicţionar să fie adus la aşa numita *formă canonică* [1] (capitolul 2). Indecşii astfel determinaţi, vor fi apoi stocaţi prin intermediul unor baze de date ne-relaţionale, precum MongoDB.

După cum a fost amintit în cadrul laboratorului nr. 4, căutarea booleană nu include mecanisme de determinare a unui eventual scor de relevanţă pentru rezultatele identificate. A treia cerinţă vă propune să studiaţi impactul *distanţei de tip cosinus* asupra rezultatelor obţinute de un motor de căutare.

2 Scurt breviar teoretic

2.1 Procesarea cuvintelor – algoritmul lui Porter (limba engleză)

Procesele prin care un cuvânt este adus la forma sa de bază poartă numele de *stemming* şi *lematizare* [1]. Cele două modele de procesare urmăresc acelaşi scop: reducerea dimensiunii indecşilor. Pentru cuvintele ce aparţin de dicţionarul unei limbi nu are sens stocarea tuturor formelor sub care aceste cuvinte pot fi derivate/conjugate (deoarece aceste transformări nu aduc efectiv informaţii noi procesului de indexare).

Modelele de analiză bazate pe *lematizare* sunt axate pe analiză morfologică şi, cel puţin teoretic, ar trebui să ofere rezultate cu un grad ridicat de precizie. Totuşi, procesele sunt lente şi considerabil mai greoi de implementat.

Procesele de tip *stemming* se bazează pe trunchierea cuvintelor (de exemplu, în limba engleză pluralul cuvintelor se obţine în majoritatea cazurilor, prin adăugarea caracterului ’s’). Aceste tehnici sunt considerabil mai simple, cu un timp de răspuns mult redus, dar nu oferă aceeaşi precizie. Unul dintre cei mai renumţi algoritmi este algoritmul lui Porter – pentru detalii vezi [2].

2.2 Căutare vectorială

Etapele de lucru implicate de această formă de căutare sunt descrise de algoritmul 1.

Algoritm 1 cautareVectoriala(D, q)

1: transforma fiecare document $d_j \in D$ in $\overrightarrow{d_j} = \{key : tf(key,d) \cdot idf(key)\}$
2: transforma interogarea $\overrightarrow{q} = \{key : tf(key,d) \cdot idf(key)\}$
3: **for all** $\overrightarrow{d_j} \in D$ **do**
4: calculeaza $s_j = similaritateCosinus(\overrightarrow{d_j}, \overrightarrow{q})$
5: **end for**
6: sorteaza documentele descrescator din punct de vedere al scorului anterior s_j
7: **return** setul relevant de documente

Observaţie: Coeficienţii *tf* şi *idf* sunt descrişi în cursul 3, slide 15, formulele (2) şi (3).

2.3 Tutorial MongoDB

NoSQL (baze de date nerelaţionale) este un model care permite memorarea unor volume imense de date, fără o structură fixă şi care se poate schimba, care oferă scalare pe orizontală [7]. MongoDB este o baza de date de documente, open-source, high performance, high availability, ce ofera scalare automata [3]. Documentul în MongoDB este o structură de date formată din perechi cheie valoare, similară cu obiectele JSON.

Exemplu stocare index direct şi invers într-o bază de date mongodb
Doc1: Data mining este o tehnica noua de analiza a datelor. Doc2: Tehnicile data mining pot aduce informatii noi. Doc3: Datele sunt colectate prin tehnici specifice.
DB indexdirectcantitativ {“doc”: “Doc1”, terms: [{“t”: “analiza”, “c”: 1}, {“t”: “data”, “c”: 2}, {“t”: “mining”, “c”: 1}, {“t”: “noutate”, “c”: 1}, {“t”: “tehnica”, “c”: 1}]} {“doc”: “Doc2”, terms: [{“t”: “data”, “c”: 1}, {“t”: “informatie”, “c”: 2}, {“t”: “mining”, “c”: 1}, {“t”: “noutate”, “c”: 1}, {“t”: “tehnica”, “c”: 1}]} {“doc”: “Doc3”, terms: [{“t”: “data”, “c”: 1}, {“t”: “tehnica”, “c”: 1}]}
DB indexinverscantitativ {“term”: “analiza”, “docs”: [{“d”: “Doc1”, “c”: 1}]} {“term”: “data”, “docs”: [{“d”: “Doc1”, “c”: 1}, {“d”: “Doc2”, “c”: 1}, {“d”: “Doc3”, “c”: 1}]} {“term”: “informatie”, “docs”: [{“d”: “Doc2”, “c”: 1}]} {“term”: “mining”, “docs”: [{“d”: “Doc1”, “c”: 1}, {“d”: “Doc2”, “c”: 1}]} {“term”: “noutate”, “docs”: [{“d”: “Doc1”, “c”: 1}, {“d”: “Doc2”, “c”: 1}]} {“term”: “tehnica”, “docs”: [{“d”: “Doc1”, “c”: 1}, {“d”: “Doc2”, “c”: 1}, {“d”: “Doc3”, “c”: 1}]}
Exemple de căutări: db.indexinverscantitativ.find({“term”:”mining”}) db.indexinverscantitativ.find({$or: [{“term”:”mining”}, {“term”:”tehnica”}])

Operaţii în mongo (inserări, căutări, actualizări): exemple pot fi consultate la adresa disponibilă în [4].
Java driver docs: [5].

```
1   MongoClient mongoClient = new MongoClient( "localhost" , 27017 );
2   MongoDatabase database = mongoClient.getDatabase("mydb");
3   MongoCollection<Document> collection = database.getCollection("test");
4   Document doc = new Document("name", "MongoDB")
5           .append("type", "database")
6           .append("count", 1)
7           .append("info", new Document("x", 203).append("y", 102));
8   collection.insertOne(doc);
9
10  Document myDoc = collection.find(eq("i", 71)).first();
11  System.out.println(myDoc.toJson());
```

Pachetul Java corespunzător: click aici.
Limbajul Python: [6].

3 Observaţii finale

1. Scenariul de test pentru proiectele voastre va fi următorul:
   (a) aplicaţiile vor primi ca mesaj de intrare numele unui director ce conţine un set de fişiere de tip *txt*;
   (b) acest director va trebui parcurs, recursiv, pentru a identifica acele fişiere *txt* şi pentru a determina cele două forme de indexare indicate;
   (c) modulul de căutare ar trebui expus printr-o pagină HTML simplă (dar nu este obligatoriu – puteţi opta pentru a dezvolta orice altă formă de interfaţă, inclusiv una de tip *command line*).
2. Încercaţi să realizaţi o **implementare paralelă/distribuită** pentru componentele ce implemeneatză modulele de indexare şi/sau cele de căutare. Ca principal model de lucru, puteţi porni de la modelul **Map & Reduce**.
3. Încercaţi să identificaţi şi să analizaţi şi alţi algoritmi de *stemming*.

Bibliografie

[1] Christopher D. Manning et. al. Introduction to Information Retrieval. http://nlp.stanford.edu/IR-book/information-retrieval-book.html, 2009.

[2] Martin Porter. The Porter stemming algorithm. http://snowball.tartarus.org/algorithms/porter/stemmer.html.

[3] mongoDB. Introduction to MongoDB. https://docs.mongodb.com/manual/introduction/.

[4] mongoDB. MongoDB CRUD Operations. https://docs.mongodb.com/manual/crud/.

[5] mongoDB. MongoDB Driver Quick Tour. http://mongodb.github.io/mongo-java-driver/3.0/driver/getting-started/quick-tour/.

[6] mongoDB. Python MongoDB Drivers. https://docs.mongodb.com/ecosystem/drivers/python/.

[7] mongoDB. What is NoSQL. https://www.mongodb.com/nosql-explained.

Barem evaluare proiect

Proiectele pornesc ca bază de notare de la **1 punct**.

Criteriu	**Punctaj**
1. Parcurgerea directorului cu setul de fişiere de indexat - *obligatoriu* parcurgerea unui singur nivel *sau* parcurgerea iterativă a unei structuri de directoare (inclusiv sub-directoarele)	 0.25 puncte **1 punct**
2. Preprocesare aplicarea (şi înţelegerea modului de funcţionare) unui algoritm de stemming pentru determinarea formelor canonice ale cuvintelor determinate	**1 punct**
3. Indexarea indirectă - *obligatoriu* (inclusiv generarea corectă a fişierlor de indecşi)	**4 puncte**
4. Căutarea în indecşii determinaţi preîncărcaţi - *obligatoriu* doar căutare booleană *sau* căutare vectorială (cu restrângerea colecţiei de lucru)	 1.5 puncte **3 puncte**

Observaţii

Studenţii vor fi rugaţi să completeze o auto-evaluare a proiectului dezvoltat. Auto-evaluările care nu diferă de nota acordată de titularii de laborator vor primi **suplimentar 0.5 puncte**. În plus, implementările pot atrage punctaj suplimentar (puncte bonus) pentru media finală a disciplinei, astfel:

stocarea indecşilor în colecţii de documente de tip MongoDB – 2 puncte;

paralelizări/distribuiri eficiente ale volumului de lucru (eventual urmărind modelul MapReduce) – până la 3 puncte.