Apache Lucene biblioteka i alati

Dragan Ivanović
dragan.ivanovic@uns.ac.rs

Katedra za informatiku, Fakultet tehničkih nauka, Novi Sad

2015.

Šta je Lucene

- Biblioteka za full-text pretraživanje
- Visoke performanse, skalabilna
- Fokus je na indeksiranju i pretraživanju
- 100% Java, ne oslanja se na druge biblioteke, nema konfiguracionih fajlova
- Nema pauka (crawler/spider), nema parsiranje posebnih formata dokumenata
- Korisnici: Wikipedia (naš diplomac!), SourceForge, TheServerSide, Nabble
- Aplikacije: Eclipse, JIRA, Nutch, Solr, Elasticsearch, Alfresco
- Open source: http://lucene.apache.org
- O. Gospodnetić, E. Hatcher. *Lucene In Action, Second edition*. Manning, 2010.

Lucene dokumenti

- Lucene indeksira i pretražuje tekstualne dokumente

Lucene dokumenti

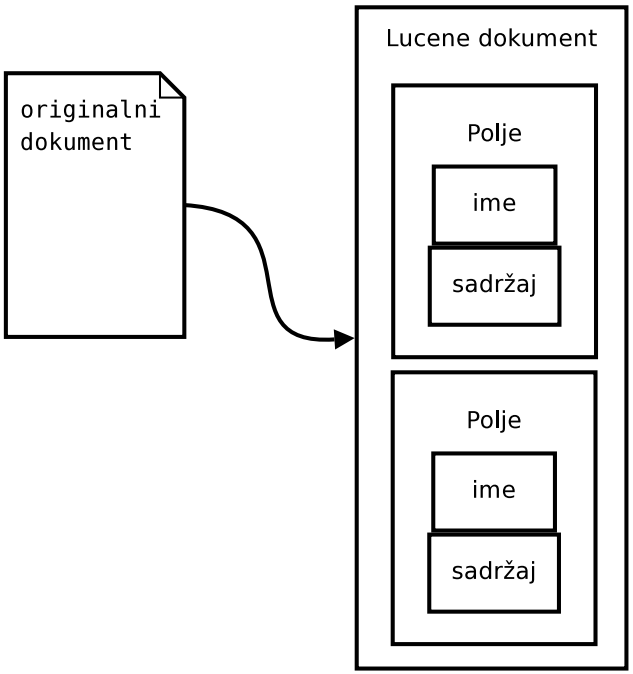
- Lucene indeksira i pretražuje tekstualne dokumente
- Dokumenti se sastoje iz polja (field)

Lucene dokumenti

- Lucene indeksira i pretražuje tekstualne dokumente
- Dokumenti se sastoje iz polja (field)
- Polja imaju ime (string) i sadržaj (string)

Lucene dokumenti

- Lucene indeksira i pretražuje tekstualne dokumente
- Dokumenti se sastoje iz polja (field)
- Polja imaju ime (string) i sadržaj (string)



Lucene indeks

- Sadržaj polja u dokumentu se procesira za invertovani indeks
- Polja mogu biti označena kao (terminologija se menja u različitim verzijama)
  - Indexed: obavezno za pretragu i sortiranje
  - Tokenized: podela na tokene, analiza teksta pre indeksiranja
  - Stored: sačuvaj originalan sadržaj polja u indeksu
  - Stored TermVectors: uz dokument sačuvan i invertovani indeks

Lucene indeks

- Sadržaj polja u dokumentu se procesira za invertovani indeks
- Polja mogu biti označena kao (terminologija se menja u različitim verzijama)
  - Indexed: obavezno za pretragu i sortiranje
  - Tokenized: podela na tokene, analiza teksta pre indeksiranja
  - Stored: sačuvaj originalan sadržaj polja u indeksu
  - Stored TermVectors: uz dokument sačuvan i invertovani indeks
- Lucene dokument može da sadrži i podatke i metapodatke!

Vrste polja

- TextField - za veće količine sadržaja, tokenized, indexed, not stored
- StringField - za identifikatore po kojima se može i pretraživati, indexed, stored, not tokenized
- StoredField - stored, not indexed, not tokenized
- LongField, IntField, DoubleField - optimizovana pretraga i sortiranje za određene vrste podataka

Indeksiranje dokumenta

```
IndexWriter writer = new IndexWriter(directory, indexWriterConfig)
Document doc = new Document();

doc.add(new TextField("author", "Angus Young",
    Store.YES));

doc.add(new TextField("author", "Malcolm Young",
    Store.YES));

doc.add(new TextField("title", "Crabsody in Blue",
    Store.YES));
// ...
writer.addDocument(doc);
writer.close();
```

Konkurentan pristup indeksu

- Najviše jedan IndexWriter može da pristupa indeksu u jednom trenutku

Konkurentan pristup indeksu

- Najviše jedan IndexWriter može da pristupa indeksu u jednom trenutku
  - nema konkurentnog indeksiranja

Konkurentan pristup indeksu

- Najviše jedan IndexWriter može da pristupa indeksu u jednom trenutku
  - nema konkurentnog indeksiranja
- Zaključavanje indeksa se vrši na nivou fajl sistema

Konkurentan pristup indeksu

- Najviše jedan IndexWriter može da pristupa indeksu u jednom trenutku
  - nema konkurentnog indeksiranja
- Zaključavanje indeksa se vrši na nivou fajl sistema
- Aktivan thread je blokiran u IndexWriter konstruktoru sve dok je indeks zauzet

Konkurentan pristup indeksu

- Najviše jedan IndexWriter može da pristupa indeksu u jednom trenutku
  - nema konkurentnog indeksiranja
- Zaključavanje indeksa se vrši na nivou fajl sistema
- Aktivan thread je blokiran u IndexWriter konstruktoru sve dok je indeks zauzet
  - jednostavna upotreba...

Konkurentan pristup indeksu

- Najviše jedan IndexWriter može da pristupa indeksu u jednom trenutku
  - nema konkurentnog indeksiranja
- Zaključavanje indeksa se vrši na nivou fajl sistema
- Aktivan thread je blokiran u IndexWriter konstruktoru sve dok je indeks zauzet
  - jednostavna upotreba...
  - ...jedan od osnovnih kriterijuma za dizajn Lucene API-ja: koristi se onako kako je najlogičnije i najjednostavnije

IndexWriter osnovne operacije

- addDocument(Document doc)
- addIndexes(IndexReader... readers)
- deleteDocuments(Term t)
- deleteDocuments(Query query)
- updateDocument(Term t, Document doc)
- close(), commit(), rollback(), flush(), getAnalyzer(), getDirectory() ...

Pretraživanje indeksa

- Upiti se mogu kostruisati na dva načina

Pretraživanje indeksa

- Upiti se mogu kostruisati na dva načina
  - pomoću jednostavnog upitnog jezika

Pretraživanje indeksa

- Upiti se mogu kostruisati na dva načina
  - pomoću jednostavnog upitnog jezika
  - programski pomoću Lucene API-ja (više mogućnosti)

Pretraživanje indeksa

- Upiti se mogu kostruisati na dva načina
  - pomoću jednostavnog upitnog jezika
  - programski pomoću Lucene API-ja (više mogućnosti)
  - (prvi način se interno svodi na drugi)

Pretraživanje indeksa

- Upiti se mogu kostruisati na dva načina
  - pomoću jednostavnog upitnog jezika
  - programski pomoću Lucene API-ja (više mogućnosti)
  - (prvi način se interno svodi na drugi)
- Moguće je pristupati indeksu radi pretraživanja iz više threadova

Pretraživanje indeksa

- Upiti se mogu kostruisati na dva načina
  - pomoću jednostavnog upitnog jezika
  - programski pomoću Lucene API-ja (više mogućnosti)
  - (prvi način se interno svodi na drugi)
- Moguće je pristupati indeksu radi pretraživanja iz više threadova
  - konkurentno pretraživanje

Pretraživanje indeksa

- Upiti se mogu kostruisati na dva načina
  - pomoću jednostavnog upitnog jezika
  - programski pomoću Lucene API-ja (više mogućnosti)
  - (prvi način se interno svodi na drugi)
- Moguće je pristupati indeksu radi pretraživanja iz više threadova
  - konkurentno pretraživanje
  - nije potreban nikakav kod za sinhronizaciju

Pretraživanje dokumenta

```
DirectoryReader reader = DirectoryReader.open(directory);
IndexSearcher searcher = new IndexSearcher(reader);

QueryParser parser = new QueryParser("contents", analyzer);
Query query = parser.parse("author:Young");

TopScoreDocCollector collector =
TopScoreDocCollector.create(hitsPerPage, true);
searcher.search(query, collector);
ScoreDoc[] hits = collector.topDocs().scoreDocs;

System.out.println("matches:" + hits.length);
Document doc = searcher.doc(0);
System.out.println("author=" + doc.get("author"));
searcher.close();
```

Upitni jezik

- Upit se izražava kao string
- Parsiranje upita obuhvata i analizu teksta
- Ne obuhvata sve postojeće tipove upita

Upitni jezik

- Upit se izražava kao string
- Parsiranje upita obuhvata i analizu teksta
- Ne obuhvata sve postojeće tipove upita
- Dva tipa termova: reči (test, hello)
  i fraze ("hello dolly")

Upitni jezik

- Upit se izražava kao string
- Parsiranje upita obuhvata i analizu teksta
- Ne obuhvata sve postojeće tipove upita
- Dva tipa termova: reči (test, hello)
  i fraze ("hello dolly")
- Term je uvek vezan za polje; polje se navodi ispred terma i razdvaja dvotačkom title:blue, author:Young

Upitni jezik

- Upit se izražava kao string
- Parsiranje upita obuhvata i analizu teksta
- Ne obuhvata sve postojeće tipove upita
- Dva tipa termova: reči (test, hello)
  i fraze ("hello dolly")
- Term je uvek vezan za polje; polje se navodi ispred terma i razdvaja dvotačkom title:blue, author:Young
- Ako se polje ne navede, podrazumeva se *default* polje
- Default polje se definiše prilikom konstrukcije parsera

Upitni jezik

- Upit se izražava kao string
- Parsiranje upita obuhvata i analizu teksta
- Ne obuhvata sve postojeće tipove upita
- Dva tipa termova: reči (test, hello)
  i fraze ("hello dolly")
- Term je uvek vezan za polje; polje se navodi ispred terma i razdvaja dvotačkom title:blue, author:Young
- Ako se polje ne navede, podrazumeva se *default* polje
- Default polje se definiše prilikom konstrukcije parsera
- Termovi se mogu povezivati logičkim operatorima
  author:Young AND title:Blue

Modifikacije terma

- Džoker znaci
  - te?t – zamenjuje jedno slovo
  - tes*, te*t – zamenjuje više slova
  - *est – džoker znak nije dozvoljen na prvom mestu!

Modifikacije terma

- Džoker znaci
  - te?t – zamenjuje jedno slovo
  - tes*, te*t – zamenjuje više slova
  - *est – džoker znak nije dozvoljen na prvom mestu!
- Fuzzy pretraga: koristi Levenshtein rastojanje (edit distance)
  - roam∼ – pronaći će i roams i foam
  - roam∼2 – minimalna edit distance je 0 (celobrojna vrednost)
  - podrazumevana edit distance je 2

Modifikacije terma

- Džoker znaci
  - te?t – zamenjuje jedno slovo
  - tes*, te*t – zamenjuje više slova
  - *est – džoker znak nije dozvoljen na prvom mestu!
- Fuzzy pretraga: koristi Levenshtein rastojanje (edit distance)
  - roam~ – pronaći će i roams i foam
  - roam~2 – minimalna edit distance je 0 (celobrojna vrednost)
  - podrazumevana edit distance je 2
- Blizinska pretraga: na kraj fraze dodati max rastojanje
  - "jakarta apache"~10 – pronađi jakarta i apache na rastojanju od max 10 reči

Modifikacije terma

- Pretraga po opsegu vrednosti
  - modDate:[20020101 TO 20030101] (zatvoreni interval)
  - title:{Aida TO Carmen} (otvoreni interval)
  - poređenje je uvek leksikografsko (npr. brojevi se neće pravilno porediti)

Modifikacije terma

- Pretraga po opsegu vrednosti
  - modDate:[20020101 TO 20030101] (zatvoreni interval)
  - title:{Aida TO Carmen} (otvoreni interval)
  - poređenje je uvek leksikografsko (npr. brojevi se neće pravilno porediti)
- Term boosting: povećavanje značaja terma
  - jakarta^4 apache – jakarta biće značajniji od apache
  - podrazumevani *boost* je 1
  - *boost* > 0

Logički operatori

- Operatori: AND, OR, NOT, +, -
- Pišu se isključivo velikim slovima

Logički operatori

- Operatori: AND, OR, NOT, +, -
- Pišu se isključivo velikim slovima
- OR je podrazumevani operator
  - jakarta apache ⇔ jakarta OR apache

Logički operatori

- Operatori: AND, OR, NOT, +, -
- Pišu se isključivo velikim slovima
- OR je podrazumevani operator
  - jakarta apache ⇔ jakarta OR apache
- AND ili && ima funkciju preseka skupova
  - jakarta AND apache

Logički operatori

- Operatori: AND, OR, NOT, +, -
- Pišu se isključivo velikim slovima
- OR je podrazumevani operator
  - jakarta apache ⇔ jakarta OR apache
- AND ili && ima funkciju preseka skupova
  - jakarta AND apache
- + zahteva postojanje terma iza sebe
  - +jakarta lucene

Logički operatori

- Operatori: AND, OR, NOT, +, -
- Pišu se isključivo velikim slovima
- OR je podrazumevani operator
  - jakarta apache ⇔ jakarta OR apache
- AND ili && ima funkciju preseka skupova
  - jakarta AND apache
- + zahteva postojanje terma iza sebe
  - +jakarta lucene
- NOT (binarni operator!) predstavlja razliku skupova
  - "jakarta apache" NOT "apache lucene"

Logički operatori

- Operatori: AND, OR, NOT, +, -
- Pišu se isključivo velikim slovima
- OR je podrazumevani operator
  - jakarta apache ⇔ jakarta OR apache
- AND ili && ima funkciju preseka skupova
  - jakarta AND apache
- + zahteva postojanje terma iza sebe
  - +jakarta lucene
- NOT (binarni operator!) predstavlja razliku skupova
  - "jakarta apache" NOT "apache lucene"
- - zahteva nepostojanje terma iza sebe
  - "jakarta apache" -lucene

Grupisanje

- Grupisanje logičkih izraza
  - (jakarta OR apache) AND website

Grupisanje

- Grupisanje logičkih izraza
  - (jakarta OR apache) AND website
- Grupisanje polja
  - title:(+return +"pink panther")

Tokenizatori

- Rastavljaju sadržaj polja na tokene
- Primer sadržaja: `full-text lucene.apache.org`
- StandardTokenizer
  - `full` `text` `lucene.apache.org`
- WhitespaceTokenizer
  - `full-text` `lucene.apache.org`
- LetterTokenizer
  - `full` `text` `lucene` `apache` `org`

Token filteri

- LowerCaseFilter – pretvara u mala slova
- StopFilter – izbacuje stop reči
- ASCIIFoldingFilter (bivši ISOLatin1AccentFilter) – uklanja akcente sa slova
- PorterStemFilter – radi Porter stemming
- SnowballFilter – drugi način za stemming
- SynonymTokenFilter – dodaje sinonime u indeks
- ...

Analizatori

- Kombinuju tokenizator i filtere
- SimpleAnalyzer
  - LetterTokenizer → LowerCaseFilter
- StandardAnalyzer
  - StandardTokenizer → StandardFilter → LowerCaseFilter → StopFilter
- CzechAnalyzer
- DutchAnalyzer
- GermanAnalyzer
- GreekAnalyzer
- RussianAnalyzer
- ...

Analizatori

- Kombinuju tokenizator i filtere
- SimpleAnalyzer
  - LetterTokenizer → LowerCaseFilter
- StandardAnalyzer
  - StandardTokenizer → StandardFilter → LowerCaseFilter → StopFilter
- CzechAnalyzer
- DutchAnalyzer
- GermanAnalyzer
- GreekAnalyzer
- RussianAnalyzer
- ...
- SerbianAnalyzer ???

SerbianAnalyzer

- Koje funkcije nam trebaju?
  - ravnopravan tretman latinice i ćirilice – upiti pronalaze tekst u oba pisma
  - mogućnost unosa upita bez akcenata – cevapcici $\Leftrightarrow$ ćevapčići
  - redukciju na koren reči cemo sada zanemariti

SerbianAnalyzer

- Koje funkcije nam trebaju?
  - ravnopravan tretman latinice i ćirilice – upiti pronalaze tekst u oba pisma
  - mogućnost unosa upita bez akcenata – cevapcici $\Leftrightarrow$ ćevapčići
  - redukciju na koren reči cemo sada zanemariti
- Ideje ?

Rangiranje dokumenata

- Koristi se kombinacija vektorskog i Bulovog modela
  - prvo se pomoću Bulovog modela izdvoje dokumenti koje treba dalje rangirati
  - ima i fuzzy proširenja
- Detalji - link

Rangiranje dokumenata

- Funkcija za ocenjivanje:

$$score(q,d) = coord(q,d) \cdot norm(q) \sum_{t \in q} tf(t \in d) \cdot idf(t) \cdot boost(t) \cdot norm(t,d)$$

  - $coord(q,d)$ – koordinacioni faktor, srazmeran broju pronađenih termova u dokumentu (search time)
  - $norm(q)$ – normalizacioni faktor koji ocene iz različitih upita čini uporedivim (search time)
  - $tf(t)$ – frekvencija terma t u dokumentu d (index time)
  - $idf(t)$ – inverzna frekvencija dokumenta za term t (index time)
  - $boost(t)$ – faktor povećanja važnosti terma t (search time)
  - $norm(t,d)$ – uključuje boost za dokument i polje, i dužinu polja (index time)

Kastomizacija

- *Analyzer* - kastomizacija pretprocesiranja teksta
- *Filter* - ne odnosi se na normalizaciju tokena, sužavanje mogućeg skupa odgovora, pretražuje se određena grupa dokumenata koji su dobijeni kao prethodni rezultat ili koji su dozvoljeni određenoj grupi korisnika (performanse mogu na ovaj način da se poprave)
- *Sort* - način sortiranja drugačiji od sortiranja po relevantnosti, na primer po nekom polju (godina, datum, itd.)
- *Collector* - dodatna obrada odgovora na upitu, može da uključi sortiranje odgovora, filtriranje pa čak i da redefiniše *score* algoritam)

Luke

- Swing aplikacija za rad sa Lucene indeks fajlovima koji su kreirani iz neke aplikacije
- Koristi se prilikom razvoja aplikacija koje koriste Lucene biblioteku
- Lista mogućnosti
  - Pregled indeksiranih zapisa po rednom broju dokumenta ili po nekom termu
  - Prikaz kompletnog dokumenta (Lucene Document) i kopiranje u *clipboard*
  - Izvršavanje upita i pregled rezultata, može se uključiti i odgovarajući *Analyzer*
  - Brisanje i izmena dokumenta, odnosno njegovih polja
  - Optimizacija indeksa
  - ...

Solr

- Platforma za pretraživanje koja se *deploy*-uje u neki veb server (Jetty, Tomcat,...)
- Open source, pisana u Javi
- Bazirana na Lucene biblioteci, ali je gotova aplikacija
- Kastomizuje se putem XML fajlova, ali ima i otvorenu arhitekturu tako da ju je moguće i proširiti ako za to ima potrebe
- Ima REST HTTP/XML i JSON API pa joj se može pristupati i iz aplikacija koje nisu pisane u Javi (SolrJ, SolrNet)
- Lista mogućnosti
  - Pretraživanje
  - Indeksiranje
  - *Highlight* rezultata
  - *Rich* dokumenti (PDF, Word, etc.) i geoprostorne pretrage
  - Dinamičko klasterovanje
  - Integracija za bazom
  - ...

Elasticsearch

- Softver otvorenog koda za pretraživanje izgrađen nad Lucene bibliotekom
- Napisan u javi,
- Stavlja akcenat na jednostavnosti, što se ogleda u jednostavnom REST API-u koji nudi
- Distribuirano skladište dokumenata gde se svako polje indeksira u real-time-u i koristi u pretrazi
- Distribuiran sistem za pretragu sa real-time analitikama
- Može da se skalira na stotine servera i da radi sa petabajtima strukturiranih i nestrukturiranih podataka
- Servisima je moguće pristupiti putem REST API-a, zatim putem web klijenta, kao i putem komandne linije

Nutch

- Konfigurabilni veb *crawler*
- Open source, pisano u Javi
- Otvorena arhitektura
- Preuzeti podaci se mogu skladištiti pomoću *Apache Gora framework*-a za skladištenje velike količine podataka
- Ekstrahovanje teksta iz dokumenta se može vršiti pomoću *Apache Tika* alata
- Indeksiranje ekstrahovanih tekstova iz dokumenata se može vršiti pomoću *Solr* ili *Elasticsearch* platforme
- Može se pokrenuti na jednoj mašini ili distribuirano u *Hadoop* klasteru

Lucene - contributions

- Pored osnovnog jezgra Lucene ima i proširenja
- Highlighter
- Geoprostorne pretrage
- Auto-suggest and Spellchecking support - Did you mean
- *Analyzer*-i za različite jezike
- *Framework* za testiranje
- ...

Snowball

- Prost jezik za obradu stringova
- String paterni odlučuju o toku izvršavanja programa
- Pogodan jezik za pisanje stemera
- Postoji konvertor napisanog stemera u *Snowball* jeziku u *Lucene Analyzer*
- Nema za srpski jezik
- Porterov stemer za engleski jezik - link