Πώς δουλεύει το Lucene

Βήμα προς βήμα στον κώδικα του IndexTester.java

Το πρόγραμμα σε δύο φάσεις

Φάση 1 — Ευρετηρίαση

Εκτελείται μόνο αν το πρώτο όρισμα είναι «1».

1. Διαβάζει το αρχείο JSON γραμμή προς γραμμή
2. Φτιάχνει ένα Document ανά γραμμή
3. Το IndexWriter το γράφει στο ευρετήριο
4. close(): αποθήκευση στον δίσκο

Φάση 2 — Αναζήτηση

Ατέρμων βρόχος, μέχρι Ctrl-C.

1. Διαβάζει ερώτημα από την κονσόλα
2. Το QueryParser το μετατρέπει σε Query
3. Το IndexSearcher βρίσκει και βαθμολογεί
4. Τυπώνει τα 10 καλύτερα αποτελέσματα

Κοινά στις δύο φάσεις: ο ίδιος **StandardAnalyzer** και ο ίδιος **Directory** (φάκελος ευρετηρίου).

Εκτέλεση και ορίσματα

```
java IndexTester 1 papers.json ./index
```

Όρισμα	Σημασία
args[0] should_clear_index	«1»: ξαναχτίζει το ευρετήριο· αλλιώς χρησιμοποιεί το υπάρχον
args[1] path_to_data	Αρχείο JSON Lines: ένα αντικείμενο JSON ανά γραμμή
args[2] path_to_index	Φάκελος όπου γράφεται ή από όπου διαβάζεται το ευρετήριο

Παράδειγμα γραμμής εισόδου:

```
{"title": "Deep Learning for Search", "abstract": "We study ..."}
```

Analyzer: από κείμενο σε όρους

```
StandardAnalyzer analyzer = new StandardAnalyzer();
```

- **Tokenizer:** σπάει το κείμενο σε λέξεις (κανόνες Unicode), αφαιρεί σημεία στίξης
- **LowerCaseFilter:** μετατρέπει όλους τους όρους σε πεζά
- Χωρίς stemming: «networks» και «network» είναι διαφορετικοί όροι
- Στις σύγχρονες εκδόσεις (Lucene 8+) δεν αφαιρεί stopwords από προεπιλογή

```
"Deep Neural Networks for Text-Mining"
→ [deep] [neural] [networks] [for] [text] [mining]
```

Κανόνας: ο ίδιος analyzer στην ευρετηρίαση και στο ερώτημα, αλλιώς οι όροι δεν ταιριάζουν.

Directory και IndexWriter

```
Directory dir = FSDirectory.open(idxDir.toPath());
IndexWriterConfig idxConfig = new IndexWriterConfig(analyzer);
idxConfig.setOpenMode(IndexWriterConfig.OpenMode.CREATE);
IndexWriter idxWriter = new IndexWriter(dir, idxConfig);
```

- **Directory:** πού ζει το ευρετήριο· FSDirectory = ένας φάκελος στον δίσκο
- **IndexWriterConfig:** οι ρυθμίσεις του writer, κυρίως ποιος analyzer θα χρησιμοποιηθεί
- **OpenMode.CREATE:** σβήνει ό,τι υπάρχει και ξεκινά από την αρχή (άλλες επιλογές: APPEND, CREATE_OR_APPEND)
- **IndexWriter:** το μόνο αντικείμενο που προσθέτει, διαγράφει ή αλλάζει έγγραφα

Document και Field

```
Map j = jParser.parse(line);
String title = (String) j.get("title");
Field tiField = new Field("title", title, TextField.TYPE_STORED);
Document thisDoc = new Document();
thisDoc.add(tiField);
idxWriter.addDocument(thisDoc);
```

Ένα **Document** είναι ένα σύνολο από **Field** (όνομα + τιμή + τύπος). Εδώ υπάρχει μόνο το πεδίο «title».

TYPE_STORED	Τι σημαίνει
indexed	Μπαίνει στο ανεστραμμένο ευρετήριο, άρα είναι αναζητήσιμο
tokenized	Περνά από τον analyzer και σπάει σε όρους
stored	Κρατιέται και το αρχικό κείμενο, για να εμφανιστεί στα αποτελέσματα

Τι χτίζεται: ανεστραμμένο ευρετήριο

Τρία έγγραφα

```
0: Deep Learning for Search
1: Search Engines in Practice
2: Deep Text Mining
```

Το docID δίνεται αυτόματα με τη σειρά του addDocument.

Για κάθε όρο, το Lucene κρατά τη λίστα των εγγράφων όπου εμφανίζεται (postings). Η αναζήτηση γίνεται ανά όρο, όχι ανά έγγραφο.

Όρος	docFreq	Postings
deep	2	0, 2
engines	1	1
for	1	0
in	1	1
learning	1	0
mining	1	2
practice	1	1
search	2	0, 1
text	1	2

Τέλος ευρετηρίασης: close()

```
idxWriter.close();
```

- Το close() κάνει **commit**: τα έγγραφα γράφονται οριστικά στον φάκελο
- Πριν από το commit, ένας νέος reader δεν βλέπει τα έγγραφα
- Το ευρετήριο αποτελείται από **segments**: μικρά, αμετάβλητα κομμάτια που αργότερα συγχωνεύονται

Ενδεικτικά αρχεία στον φάκελο path_to_index:

```
segments_1 _0.cfs _0.cfe _0.si write.lock
```

Άνοιγμα του ευρετηρίου για αναζήτηση

```
DirectoryReader reader = DirectoryReader.open(dir);
IndexSearcher searcher = new IndexSearcher(reader);
QueryParser qParser = new QueryParser("title", analyzer);
```

- **DirectoryReader:** στιγμιότυπο (snapshot) του ευρετηρίου, μόνο για ανάγνωση
- **IndexSearcher:** εκτελεί τα ερωτήματα πάνω στον reader και βαθμολογεί τα έγγραφα
- **QueryParser:** μετατρέπει το κείμενο του χρήστη σε αντικείμενο Query
- Το «title» είναι το **προεπιλεγμένο πεδίο**: εκεί ψάχνει όταν το ερώτημα δεν ορίζει πεδίο
- Ο ίδιος analyzer με την ευρετηρίαση: «Search» γίνεται «search» και ταιριάζει

Σύνταξη ερωτημάτων του QueryParser

Ερώτημα	Σημασία
search engines	search OR engines (η προεπιλογή είναι OR)
search AND engines	Και οι δύο όροι υποχρεωτικά
+search +deep	Το ίδιο, με τελεστές + (υποχρεωτικός όρος)
search -deep	Περιέχει search αλλά όχι deep
"text mining"	Φράση: οι όροι διαδοχικά
eng*	Πρόθεμα (wildcard): engine, engines, …
serch~	Ασαφής (fuzzy) αναζήτηση: ανέχεται ορθογραφικά λάθη
title:search	Ρητό πεδίο (εδώ υπάρχει μόνο το title)

Εκτέλεση ερωτήματος και αποτελέσματα

```
Query q = qParser.parse(queryText);
TopDocs results = searcher.search(q, 10);
System.out.println("Got " + results.totalHits);
for (ScoreDoc d : results.scoreDocs) {
    System.out.println(d.doc + "\t" + d.score);
    Document res = searcher.storedFields().document(d.doc);
    System.out.println(res.getField("title").stringValue());
}
```

- **search(q, 10):** επιστρέφει τα 10 έγγραφα με τον υψηλότερο βαθμό
- **totalHits:** πόσα έγγραφα ταίριαξαν συνολικά (όχι μόνο τα 10)
- **ScoreDoc:** το docID και ο βαθμός (score) κάθε αποτελέσματος
- **storedFields():** φέρνει τον αποθηκευμένο τίτλο· δουλεύει μόνο επειδή το πεδίο είναι TYPE_STORED

Από πού προκύπτει το score: BM25

Από προεπιλογή το Lucene βαθμολογεί με **BM25**. Για κάθε όρο του ερωτήματος που υπάρχει στο έγγραφο, ο βαθμός αυξάνεται όταν:

Παράγοντας	**Επίδραση**
IDF	Ο όρος είναι σπάνιος στη συλλογή (μικρό docFreq)
TF	Ο όρος εμφανίζεται πολλές φορές στο πεδίο, με κορεσμό
Μήκος πεδίου	Ο τίτλος είναι σύντομος σε σχέση με τον μέσο όρο

Οι βαθμοί των όρων αθροίζονται. Το searcher.explain(q, docID) δείχνει αναλυτικά τον υπολογισμό.