

## TDT4117 - Øving 3

Ole Christer Selvig, Håkon Løvdal og Kristoffer Andreas Dalby

29. oktober 2013

## Oppgave 1 - Interpolated Precision

## Oppgave 2 - Lucene

### Hva er Lucene

Lucene er et rammeverk/bibliotek for informasjons gjenfinning. Det er skrevet i java av Doug Cutting og er idag blitt et veldig omfattende prosjekt som er mye brukt for å bygge applikasjoner for informasjonsgjenfinning. Kjerne konseptet med Lucene er at et dokument inneholder mange felter med tekst. Dette gjør at Lucene kan brukes på mange forskjellige filtyper, altså alle filer hvor det er mulighet å hente ut tekst. Lucene er også skrevet om til flere andre programmeringsspråk som python og C++.

### Indeksering med Apache Lucene

I eksempelprogrammet vi har fått utdelt starter selve indekseringsprosessen i metoden `indexDocs`. Før denne metoden blir kalt kjøres hovedsaklig lesing og forberedelse av filen. I `indexDocs` metoden kjøres det først en sjekk om parameteret som mottas er en fil eller en mappe. Hvis det er en mappe vil metoden rekursivt på seg selv over alle filene i mappen.

Når funksjonen så får en fil vil den begynne indekseringsprosessen. Dette starter med:

```
System.out.println("Indexing to directory '" + indexPath + "'...");
Directory dir = FSDirectory.open(new File(indexPath));
Analyzer analyzer = new StandardAnalyzer(Version.LUCENE_31);
IndexWriterConfig iwc = new IndexWriterConfig(Version.LUCENE_31,
    analyzer);
```

Det første som skjer når programmet starter etter det har tatt inn alle input parametrene er å gi output til brukeren om hva som skal analyseres. Deretter starter programmet med å instansiere et nytt `Directory` objekt som holder styr på hvilket direktiv som skal indekseres. Når dette er gjort vil den lage en ny analyzer som instansierer `StandardAnalyzer` med Lucene versjonen vi bruker. Denne sendes inn som et parameter i `IndexWriterConfig` som holder all konfigurasjonen for `IndexWriter`en som blir laget senere.

```
if (create) {
    iwc.setOpenMode(OpenMode.CREATE);
} else {
    iwc.setOpenMode(OpenMode.CREATE_OR_APPEND);
}
```

For å konfigurere `IndexWriter`en sjekker vi om det ble sendt med et parameter om å lage en ny indeks eller oppdatere en gammel.

```
IndexWriter writer = new IndexWriter(dir, iwc);
indexDocs(writer, docDir);
```

Til slutt før selve indekseringen startes opprettes en `IndexWriter`, denne har som oppgave å lage og håndtere/vedlikeholde en indeks. Som argument blir direktivet og konfigurasjonen som ble laget tidligere lagt med.

```
Document doc = new Document();
```

Her instansieres et nytt dokument som skal representere filen ferdig indeksert.

```
Field pathField = new Field("path", file.getPath(), Field.Store.YES
    , Field.Index.NOT_ANALYZED_NO_NORMS);
pathField.setIndexOptions(IndexOptions.DOCS_ONLY);
doc.add(pathField);
```

Deretter begynner man å fylle filen med data om filen, denne kodebiten legger inn filstien fra originalfilen til et felt i det nye, indekserte dokumentet.

```
NumericField modifiedField = new NumericField("modified");
modifiedField.setLongValue(file.lastModified());
doc.add(modifiedField);
```

Det vil også bli lagt til et felt hvor det lagres når den originale filen ble sist modifisert.

```
doc.add(new Field("contents", new BufferedReader(new
    InputStreamReader(fis, "UTF-8"))));
```

Til slutt i indekseringsprosessen vil filen bli åpnet i en leser for å indeksere filen og legge resultatet til som et felt i dokumentet. Her blir også UTF-8 valgt får å korrekt håndtere spesialtegn.

```
if (writer.getConfig().getOpenMode() == OpenMode.CREATE) {
    // New index, so we just add the document (no old
    // document can be there):
    System.out.println("adding_" + file);
    writer.addDocument(doc);
} else {
    // Existing index (an old copy of this document may have
    // been indexed) so
    // we use updateDocument instead to replace the old one
    // matching the exact
    // path, if present:
    System.out.println("updating_" + file);
    writer.updateDocument(new Term("path", file.getPath()),
        doc);
}
```

Til slutt vil den indekserte dokumentet bli skrevet til et nytt indeksert dokument. Det kan som vist også bli oppdatert om filen allerede finnes fra før.

```
Indexing to directory 'index'...
adding /home/kradalby/workspace/ir_assignment3/src/apple/doc10.txt
adding /home/kradalby/workspace/ir_assignment3/src/apple/doc1.txt
adding /home/kradalby/workspace/ir_assignment3/src/apple/doc6.txt
adding /home/kradalby/workspace/ir_assignment3/src/apple/doc8.txt
adding /home/kradalby/workspace/ir_assignment3/src/apple/doc4.txt
adding /home/kradalby/workspace/ir_assignment3/src/apple/doc9.txt
adding /home/kradalby/workspace/ir_assignment3/src/apple/doc2.txt
adding /home/kradalby/workspace/ir_assignment3/src/apple/doc5.txt
adding /home/kradalby/workspace/ir_assignment3/src/apple/doc7.txt
adding /home/kradalby/workspace/ir_assignment3/src/apple/doc3.txt
378 total milliseconds
```

## Søk med Apache Lucene

Søkeord: *apple*

```
Enter query:
apple
Searching for: apple
7 total matching documents
1. /home/kradalby/workspace/ir_assignment3/src/apple/doc7.txt
2. /home/kradalby/workspace/ir_assignment3/src/apple/doc2.txt
3. /home/kradalby/workspace/ir_assignment3/src/apple/doc4.txt
4. /home/kradalby/workspace/ir_assignment3/src/apple/doc10.txt
5. /home/kradalby/workspace/ir_assignment3/src/apple/doc1.txt
6. /home/kradalby/workspace/ir_assignment3/src/apple/doc3.txt
7. /home/kradalby/workspace/ir_assignment3/src/apple/doc5.txt
Press (q)uit or enter number to jump to a page.
```

Søkeord: *apple grape*

```
Enter query:
apple grape
Searching for: apple grape
8 total matching documents
1. /home/kradalby/workspace/ir_assignment3/src/apple/doc2.txt
2. /home/kradalby/workspace/ir_assignment3/src/apple/doc10.txt
3. /home/kradalby/workspace/ir_assignment3/src/apple/doc3.txt
4. /home/kradalby/workspace/ir_assignment3/src/apple/doc5.txt
5. /home/kradalby/workspace/ir_assignment3/src/apple/doc9.txt
6. /home/kradalby/workspace/ir_assignment3/src/apple/doc7.txt
7. /home/kradalby/workspace/ir_assignment3/src/apple/doc4.txt
8. /home/kradalby/workspace/ir_assignment3/src/apple/doc1.txt
Press (q)uit or enter number to jump to a page.
```

Søkeord: *apple grape melon*

```
Enter query:
apple grape melon
Searching for: apple grape melon
10 total matching documents
1. /home/kradalby/workspace/ir_assignment3/src/apple/doc10.txt
2. /home/kradalby/workspace/ir_assignment3/src/apple/doc3.txt
3. /home/kradalby/workspace/ir_assignment3/src/apple/doc5.txt
4. /home/kradalby/workspace/ir_assignment3/src/apple/doc2.txt
5. /home/kradalby/workspace/ir_assignment3/src/apple/doc9.txt
6. /home/kradalby/workspace/ir_assignment3/src/apple/doc1.txt
7. /home/kradalby/workspace/ir_assignment3/src/apple/doc4.txt
8. /home/kradalby/workspace/ir_assignment3/src/apple/doc7.txt
9. /home/kradalby/workspace/ir_assignment3/src/apple/doc6.txt
10. /home/kradalby/workspace/ir_assignment3/src/apple/doc8.txt
Press (q)uit or enter number to jump to a page.
```

I disse søkene blir booleanmodellen brukt som querymodell, da de returnerte dokumentene, altså logisk ELLER.

## Oppgave 3 - Lucene's Boolean Query Model

Den utdelte koden vil først instansiere et *BooleanQuery*-objekt som kan representere en boolsk konjunktiv normalform på formen:

$$(V_1 \wedge \dots \wedge V_n)$$

Det neste den gjør er å instansiere et *StringTokenizer*-objekt som deler opp spørringen i single termer og har en iterator, som gjør det lett å iterere over alle termene.

Deretter vil *while-loopen* legge til en *TermQuery* til den boolske spørringen, med en boolsk *MUST*-kondisjon på hver term, noe som tilsvarer *logisk OG*. Med dette bygges det en *BooleanQuery* som representerer en boolsk spørring på konjunktiv normalform. Så svaret på hvordan boolsk spørring som prosesseres med denne koden er:

*BooleanClause.Occur.MUST* er det samme som *AND*

Resultater ved å kjøre med *MUST*-clause:

Søkeord: *apple*

```
Searching for: +apple
7 total matching documents
1. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc7.txt
2. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc2.txt
3. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc4.txt
4. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc10.txt
5. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc1.txt
6. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc3.txt
7. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc5.txt
```

Søkeord: *apple, grape*

```
Searching for: +apple +grape
4 total matching documents
1. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc2.txt
2. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc10.txt
3. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc3.txt
4. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc5.txt
```

Søkeord: *apple, grape, melon*

```
Searching for: +apple +grape +melon
3 total matching documents
1. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc10.txt
2. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc3.txt
3. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc5.txt
```

Videre, for å kunne gjennomføre et *logisk ELLER*-søk så må vi endre litt på den utdelte koden. For å tilfredstille et *ELLER*-søk må:

*BooleanClause.Occur.MUST* byttes ut med *BooleanClause.Occur.SHOULD*

Koden vil da se slik ut:

```
BooleanQuery query = new BooleanQuery();
StringTokenizer st = new StringTokenizer(line);
    while (st.hasMoreTokens()) {
        TermQuery tq = new TermQuery(new Term("contents",
            st.nextToken()));
        query.add(tq, BooleanClause.Occur.SHOULD);
    }
```

## Resultater ved å kjøre med SHOULD-clause

Søkeord: *apple*

```
Searching for: apple
7 total matching documents
1. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc7.txt
2. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc2.txt
3. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc4.txt
4. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc10.txt
5. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc1.txt
6. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc3.txt
7. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc5.txt
```

Søkeord: *apple, grape*

```
Searching for: apple grape
8 total matching documents
1. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc2.txt
2. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc10.txt
3. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc3.txt
4. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc5.txt
5. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc9.txt
6. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc7.txt
7. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc4.txt
8. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc1.txt
```

Søkeord: *apple, grape, melon*

```
Searching for: apple grape melon
10 total matching documents
1. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc10.txt
2. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc3.txt
3. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc5.txt
4. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc2.txt
5. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc9.txt
6. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc1.txt
7. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc4.txt
8. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc7.txt
9. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc6.txt
10. /Users/hakloev/git/TDT4117/oving3/java/src/apple/doc8.txt
```