Indizierung – Lucene

Übersicht

- Grundidee des Indexing
- Lucene – Wichtige Methoden und Klassen
- Lucene – Indizierungsbeispiele
- Lucene – Suchbeispiele
- Lucene – QueryParser Syntax

Grundideen und Ziel des Indexing

Effizientes Suchen in „großer“ Menge von Dokumenten.

Beispiel Suchmaschine

Ergebnisse **1** - **10** von ungefähr **27.600.000** für **Indexing**. (**0,04** Sekunden)

Nach eigenen Angaben hat Google 1 000 000 000 000 (eine Billion) Seiten indiziert. (Stand: 25. 7. 2008)

Naives Vorgehen

Bei Suchanfrage alle Dokumente durchsuchen.

Problem: Viel zu langsam!

Daher: **Vorverarbeiten** der zu durchsuchenden Dokumente!

Tokenizing & Normalisierung

Tokenizing: Konvertieren der Dokumente in Liste von **semantischen Einheiten**, sog. **Token**.
Dies können z.B. einzelne Worte sein.
Entfernen von Irrelevantem, z.B. Satzzeichen.

Normalisieren: Token werden auf **standardisierte Form** gebracht;
Beispielsweise: Groß-/Kleinschreibung, Singular/Plural, etc.

Die normalisierten Token heißen **Terme**.

Inzidenzmatrix

Idee: Erstelle Matrix A = (a_ij) mit binären Einträgen.
Die **Spalten** von A stehen für die **Dokumente**,
die **Zeilen** von A stehen für die **Terme**.
a_ij = 1 :<=> Term i kommt in Dokument j vor.
a_ij = 0 :<=> Term i kommt nicht in Dokument j vor.

Damit kann man **schneller suchen**.
Man hat aber **viel Overhead** wegen im Allgemeinen
vieler Nullen (insbesondere **hoher Speicherplatzbedarf**)!

Inverted Index

Speichere Liste aller Terme.
Für jeden Term t speichere Liste aller Dokumente,
in denen t vorkommt.

Bei Suchanfrage nach Term t gebe
die Dokument-Liste von t zurück.

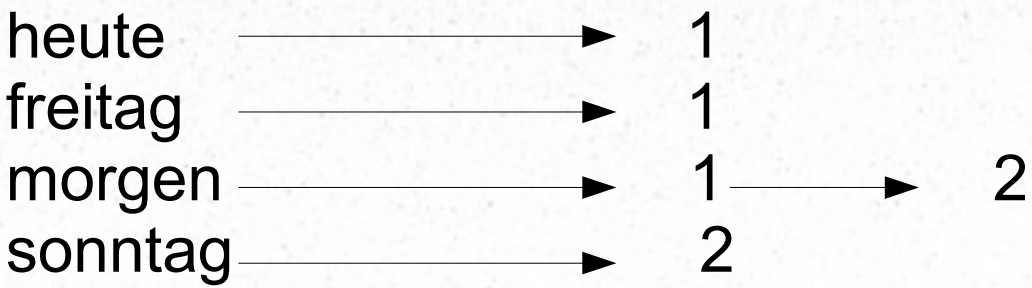
Vorteil gegenüber Inzidenzmatrix:
Im Allgemeinen erheblich weniger Overhead,
da die „Nullen“ nicht gespeichert
und nicht abgerufen werden.

Beispiel

Dokument 1: „Heute ist Freitag, und nicht morgen.“
Dokument 2: „Morgen ist Sonntag.“

Terme: heute freitag morgen sonntag

Inverted Index:



Lucene

- Java-Bibliothek zum Erstellen und Durchsuchen von Indizes
- Auf Delphi, Perl, C#, C++, Python, Ruby u.a. portiert
- Unabhängig vom Dateiformat
- Hohe Performance und Skalierbarkeit

Document

- die Einheit des Suchens und Indizierens
- ist eine Menge von Fields
- ein Field besitzt einen Namen und Textinhalt

  z.B. Name: "title"
  Inhalt: "Indexing mit Lucene"

Analyzer

- konvertiert Text zu Tokens
- die abstrakte Methode tokenStream(fieldName, reader) ist zu implementieren
- Implementationen: StandardAnalyzer, GermanAnalyzer, ...

IndexWriter

- erstellt und ändert den Index
- Hinzufügen von Dokumenten mit addDocument(doc)
- benutzt einen Analyzer, um den Inhalt der Documents zu tokenisieren

IndexSearcher

- durchsucht einen Index
- search(query, n) → liefert die besten n Dokumente

Query

- TermQuery: Suchanfrage nach Dokumenten, die einen Term enthalten
- BooleanQuery: Boolesche Verknüpfung von Querys
- kann durch QueryParser erzeugt werden

QueryParser

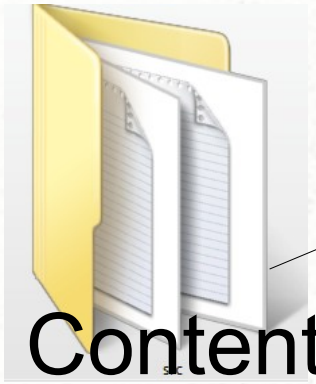
- parse(String) erstellt ein Query aus einer Suchanfrage

- Eine Suchanfrage besteht aus Termen und Operatoren

- Terme sind einzelne Wörter oder Phrasen (Text innerhalb von Anführungszeichen)

- Operatoren: AND, OR, NOT, +, -, :

Indizierung von Inhalten



Content

Plaintext

Dies ist ein beliebiger Text

<IndexFiles.class>

Index

Inverted Index

dies: d1
beliebiger: d1
text: d1

Verarbeitung von Plaintext

Plaintext

Dies ist ein beliebiger Text

StandardAnalyzer

dies ~~ist ein~~ beliebiger **t**ext

Hier:
GermanAnalyzer

IndexWriter

dies: d1
beliebiger: d1
text: d1

Index

Indizierung durchführen

Beispiel: Indizierung des „src“ - Verzeichnisses von lucene

In das Verzeichnis „lucene“ wechseln

```
C:\Users\nedunk>cd Projekte\X000\lucene-2.4.0
```

Classpath setzen

```
\X000\lucene-2.4.0>set CLASSPATH=lucene-core-2.4.0.jar;lucene-demos-2.4.0.jar
```

Lucene Kern

Lucene Demo Paket mit eignen Indizierungsregeln

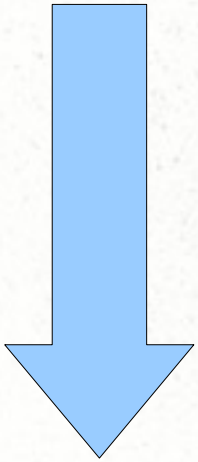
Indizierung durchführen

Verzeichnis „src“ indizieren

```
C:\Users\nedunk\Projekte\X000\lucene-2.4.0>java org.apache.lucene.demo.IndexFiles src
Indexing to directory 'index'...
adding src\demo\org\apache\lucene\demo\DeleteFiles.java
adding src\demo\org\apache\lucene\demo\FileDocument.java
adding src\demo\org\apache\lucene\demo\html\Entities.java
adding src\demo\org\apache\lucene\demo\html\HTMLParser.java
adding src\demo\org\apache\lucene\demo\html\HTMLParser.jj
adding src\demo\org\apache\lucene\demo\html\HTMLParserConstants.java
adding src\demo\org\apache\lucene\demo\html\HTMLParserTokenManager.java
adding src\demo\org\apache\lucene\demo\html\ParseException.java
adding src\demo\org\apache\lucene\demo\html\ParserThread.java
adding src\demo\org\apache\lucene\demo\html\SimpleCharStream.java
adding src\demo\org\apache\lucene\demo\html\Tags.java
adding src\demo\org\apache\lucene\demo\html\Test.java
adding src\demo\org\apache\lucene\demo\html\Token.java
adding src\demo\org\apache\lucene\demo\html\TokenMgrError.java
adding src\demo\org\apache\lucene\demo\HTMLDocument.java
adding src\demo\org\apache\lucene\demo\IndexFiles.java
adding src\demo\org\apache\lucene\demo\IndexHTML.java
adding src\demo\org\apache\lucene\demo\SearchFiles.java
adding src\jsp\configuration.jsp
adding src\jsp\footer.jsp
adding src\jsp\header.jsp
adding src\jsp\index.jsp
adding src\jsp\README.txt
adding src\jsp\results.jsp
adding src\jsp\WEB-INF\web.xml
Optimizing...
2338 total milliseconds
```

Indizierte Inhalte durchsuchen

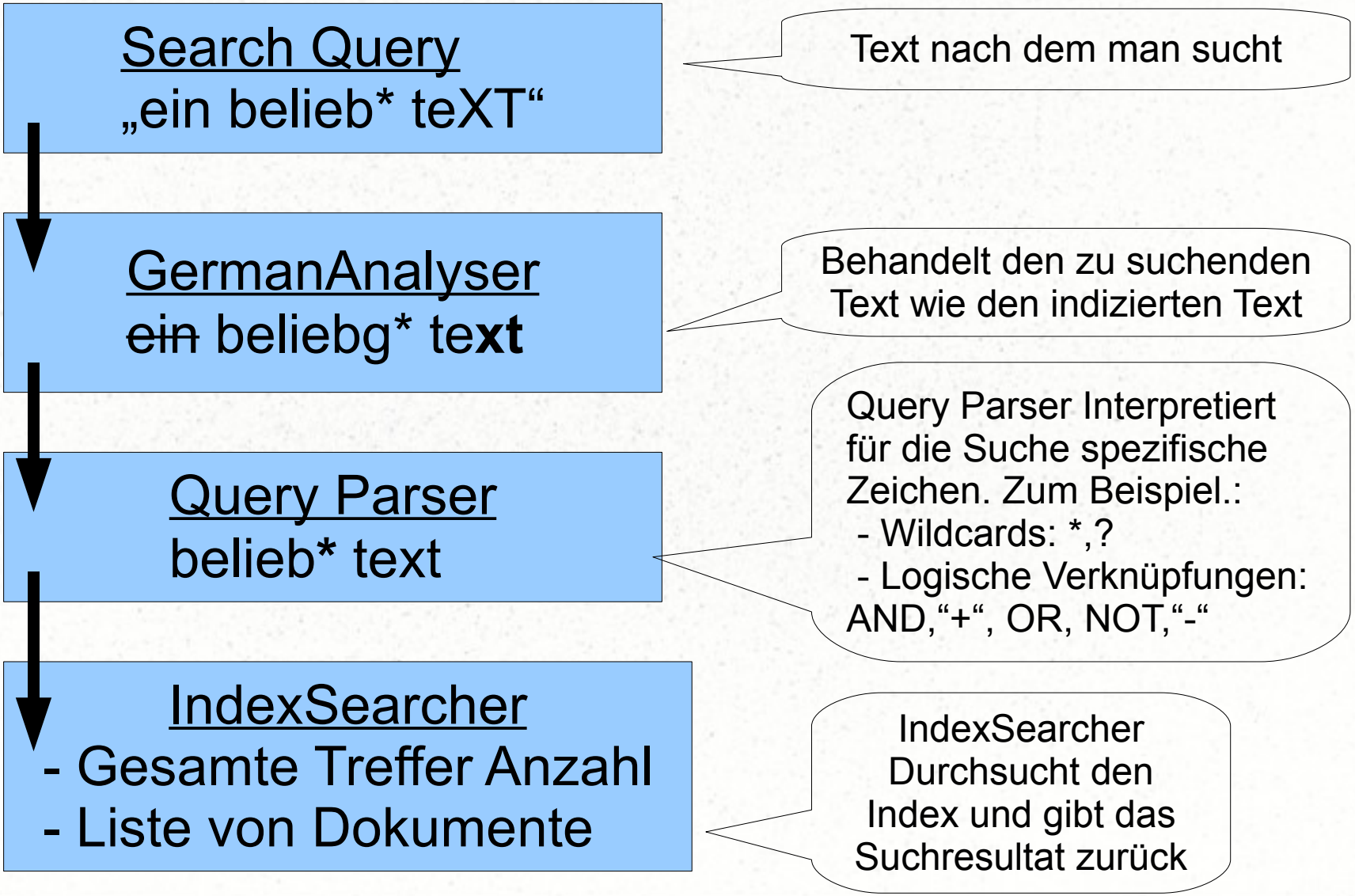
Suche nach: „ein belieb* teXT“



<SearchFiles.class>

Search Results
- Gesamte Treffer Anzahl
- Liste der Dokumente mit den gesuchten Inhalten

Indizierte Inhalte durchsuchen



Indizierte Inhalte durchsuchen

Suche nach „hanz franz“: keine Ergebnisse
Suche nach „java doc“: Implizierte „java OR doc“ Suche

```
C:\Users\nedunk\Projekte\X000\lucene-2.4.0>java org.apache.lucene.demo.SearchFiles
Enter query:
hanz franz
Searching for: hanz franz
0 total matching documents
Press (q)uit or enter number to jump to a page.

Enter query:
java doc
Searching for: java doc
8 total matching documents
1. src\jsp\results.jsp
2. src\jsp\README.txt
3. src\demo\org\apache\lucene\demo\DeleteFiles.java
4. src\demo\org\apache\lucene\demo\FileDocument.java
5. src\demo\org\apache\lucene\demo\IndexHTML.java
6. src\demo\org\apache\lucene\demo\IndexFiles.java
7. src\demo\org\apache\lucene\demo\SearchFiles.java
8. src\demo\org\apache\lucene\demo\HTMLDocument.java
Press (q)uit or enter number to jump to a page.
```

Indizierte Inhalte durchsuchen

- Verwendung von booleschen Operator AND
- Gleiche Resultate für die Suchanfragen „java AND doc“ und „JAVa and doC“

```
Enter query:
doc
Searching for: doc
6 total matching documents
1. src\jsp\README.txt
2. src\demo\org\apache\lucene\demo\FileDocument.java
3. src\demo\org\apache\lucene\demo\IndexHTML.java
4. src\demo\org\apache\lucene\demo\SearchFiles.java
5. src\demo\org\apache\lucene\demo\HTMLDocument.java
6. src\jsp\results.jsp
Press (q)uit or enter number to jump to a page.

Enter query:
java AND doc
Searching for: +java +doc
1 total matching documents
1. src\jsp\results.jsp
Press (q)uit or enter number to jump to a page.

Enter query:
JAVa AND doC
Searching for: +java +doc
1 total matching documents
1. src\jsp\results.jsp
Press (q)uit or enter number to jump to a page.
```

Query Parser Syntax

LucenePackage.tar.gz\docs\queryparsersyntax.html

- **Terms**
- **Fields**
- **Term Modifiers**
  - **Wildcard Searches**
  - **Fuzzy Searches**
  - **Proximity Searches**
  - **Range Searches**
  - **Boosting a Term**
- **Boolean Operatros**
- **Grouping**

Terms

Einzelne Wörter wie:

haus, maus, santaklaus

Fields

Metadaten und Inhaltsinformationen:

Z.B.: title, text, mod_time

Verwendung:
title: „MS DOS for Advanced Users“

Term Modifiers

Wildcard Searches: ?,*

Suche nach: te**?**t
Findet: test, text

Suche nach: test*
Findet: test, tests oder testers

Fuzzy Searches: ~

Suche nach: haus~
Findet: raus, maus, hausen

Term Modifiers

Proximity Searches: „term1 term2“~[digit]

Suche nach: „java insel“~5
Findet: „Java ist eine Insel“
Aber nicht: „Java ist eine sehr interessante Insel“

Range Searches: field:[begin to end]

Suche nach: mod_date:[20080101 to 20080514]
Findet. Alles in diesem Zeitraum

Term Modifiers

Boosting a Term: ^[digit]

Suche nach: python^4 java
Bewertet die Dokumente mit „python“ **wichtiger** als mit „java“

Somit:
Dokumente mit „python“ haben eine **höhere Relevanz**

Boolean Operators

AND, „+“, OR, NOT, „-“
Default: OR

Suche nach: java doc
Ergebnisse für java **oder** doc

Suche nach:java **AND** doc (+java +doc)
Ergebnisse in dem java **und** doc vorhanden sind

Suche nach: java **NOT** doc (java -doc)
Ergebnisse in denen java aber **nicht** doc vorkommt

Grouping

Suche nach: (Dokumentation OR Beispiel) AND Python
Findet: Dokumente mit „Python“ **und** „Dokumentation“ **oder** mit „Python“ **und** „Beispiel“

Field Grouping

Suche nach: title:(+“TurboGears 2“ +python)
Findet: Dokumente mit **dem Satz** „TurboGears 2“ **und dem Wort** „python“ in Feld **title**

Danke für die Aufmerksamkeit!