

CS520 Web Programming  
Full Text Search

Chengyu Sun  
California State University, Los Angeles

## Search Text

- ◆ Web search
- ◆ Desktop search
- ◆ Applications
  - Search posts in a bulletin board
  - Search product descriptions at an online retailer
  - ...

## Database Query

◆ Find the posts regarding "SSHD login errors".

```
select * from posts  
where content like '%SSHD login errors%';
```

Here are the steps to take to fix the SSHD login errors:  
...

Please help! I got SSHD login errors!

## Problems with Database Queries

Please help! I got an error when I tried to login through SSHD!

There a problem recently discovered regarding SSHD and login. The error message is usually ...

The solution for sshd/login errors: ...

◆ And how about performance??

## Full Text Search (FTS)

- ◆ More formally known as Information Retrieval (IR)
- ◆ Search LARGE amount of textual data

## Characteristics of FTS

◆ Vs. Databases

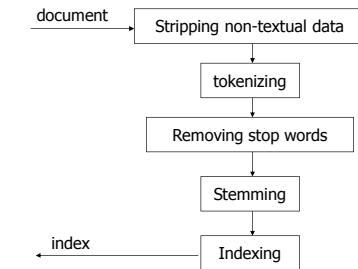
- "Fuzzy" query processing
- Relevancy ranking

## Accuracy of FTS

$$\text{Precision} = \frac{\# \text{ of relevant documents retrieved}}{\# \text{ of documents retrieved}}$$

$$\text{Recall} = \frac{\# \text{ of relevant documents retrieved}}{\# \text{ of relevant documents}}$$

## Journey of a Document



## Document

### ❖ Original

```
<html>
<body>
  <p>The solution for
  sshd/login errors:
  ...</p>
</body>
</html>
```

### ❖ Text-only

The solution for  
sshd/login errors:  
...

## Tokenizing

[the] [solution] [for] [sshd] [login] [errors]

...

## Chinese Text Example

Text: 今天天气不错。

Unigram:

[今] [天] [天] [气] [不] [错]

Bigram:

[今天] [天天] [天气] [气不] [不错]

Grammar-based:

[今天] [天气] [不错]

## Stop Words

❖Words that do not help in search and retrieval

- Function words: a, an, and, the, of, for ...

❖After stop words removal:

[ ] [solution] [ ] [sshd] [login] [errors]

...

*Problem of stop word removal??*

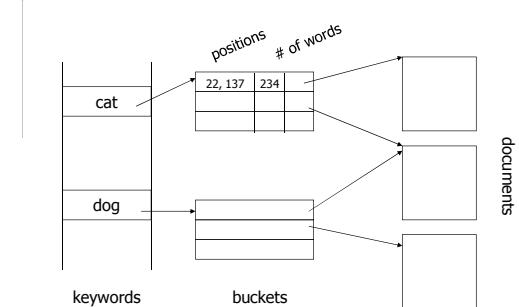
## Stemming

- ◆ Reduce a word to its stem or root form.
- ◆ Examples:

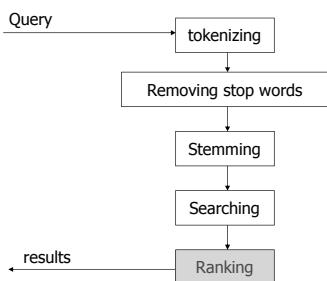
connection, connections  
connected, connecting → connect  
connective

[solution] [sshd] [login] [errors] → [solve] [sshd] [login] [error]  
...  
...

## Inverted Index



## Query Processing



## Ranking

- ◆ How well the document matches the query
  - E.g. weighted vector distance
- ◆ How "important" the document is
  - E.g. based on ratings, citations, and links

## FTS Implementations

- ◆ Databases
  - MySQL: MyISAM tables only
  - PostgreSQL (since 8.3)
  - Oracle, DB2, MS SQL Server, ...
- ◆ Standard-alone IR libraries
  - Lucene, Egothor, Xapian, MG4J, ...

## FTS from the Perspective of Application Developers

- ◆ Prepare data
- ◆ Create query
- ◆ Display result
- ◆ (Index)
- ◆ (Ranking)

## Lucene Overview

- ◆ <http://lucene.apache.org/>
- ◆ Originally developed by Doug Cutting
- ◆ THE full text search solution for Java applications
- ◆ Handles text only – needs external converters to convert other document types to text
- ◆ Java API -  
[http://lucene.apache.org/java/3\\_4\\_0/api/core/index.html](http://lucene.apache.org/java/3_4_0/api/core/index.html)

## Example 1: Index Text Files

- ◆ Directory
- ◆ Document and Field
- ◆ Analyzer
- ◆ IndexWriter

## Directory

- ◆ A place where the index files will be stored
- ◆ FSDirectory – file system directory
- ◆ RAMDirectory – virtual directory in memory

## Analyzer

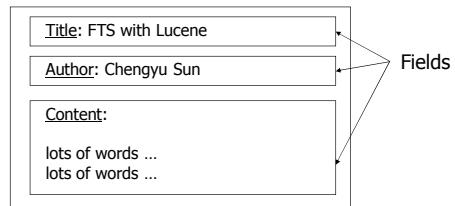
- ◆ Pre-processing the document or query text – tokenization, stop words removal, stemming ...
- ◆ Lucene built-in analyzers
  - WhitespaceAnalyzer, SimpleAnalyzer, StopAnalyzer
  - StandardAnalyzer
    - Grammar-based
    - Recognize special tokens such as email addresses
    - Handle CJK text

## IndexWriter

- ◆ addDocument( Document )
- ◆ close()
- ◆ optimize()

## Document

- ◆ A document consists of a number of user-defined fields



## Types of Fields

- ◆ Indexed – whether the field is indexed
  - Analyzed
  - Not analyzed
- ◆ Stored – whether the original text is stored together with the index

## Common Usage of Field Types

| Field                   | Indexed | Analyzed | Stored |
|-------------------------|---------|----------|--------|
| String                  | Y       | Y        | Y      |
| Large text file         | Y       | Y        |        |
| ID, people's name, date | Y       |          | Y      |
| Non-searchable data     |         |          | Y      |

## Example 2: Search

- ◆ `Query` and `QueryParser`
- ◆ `IndexSearcher`
- ◆ `TopDocs` and `ScoreDoc`
- ◆ `Document` (again)

## Queries

full text search  
+full +text search  
+full +text -search  
+title:"text search"  
+(title:full title:text) -author:"john doe"

## IndexSearcher

- ◆ `search( Query, int n )` – returns the top n results for the query

## TopDocs and ScoreDoc

- ◆ `TopDocs` contains an array of `ScoreDoc`, which has a *document id* and the *relevancy score* of the document

## Factors in Lucene Score

- ❖ # of times a term appears in a document
- ❖ # of documents that contain the term
- ❖ # of query terms found
- ❖ length of a field
- ❖ boost factor - field and/or document
- ❖ query normalizing factor – does not affect ranking

*See the API documentation for the Similarity class.*

## Document (again)

- ❖ Methods to retrieve data stored in the document
  - String get( String fieldName )

## Handle Rich Text Documents

- ❖ HTML
  - NekoHTML
- ❖ PDF
  - PDFBox
- ❖ MS Word
  - POI
- ❖ More at Lucence FAQ -  
<http://wiki.apache.org/jakarta-lucene/LuceneFAQ>

## Further Readings

- ❖ *Lucene in Action* (2ed Ed) by Michael McCandless, Erik Hatcher and Otis Gospodnetic

## FTS in PostgreSQL

- ❖ Since 8.3
  - tsearch/tsearch2 module before 8.3
- ❖ <http://www.postgresql.org/docs/9.3/intro/textsearch.html>

## Text Search Configuration

- ❖ Specify the options to transform a document to a `tsvector` – tokenization, stop words removal, stemming etc.
- ❖ psql commands
  - \dF
  - show default\_text\_search\_config;
  - set default\_text\_search\_config=english;
- ❖ Change default text search configuration in  
`$DATA/postgresql.conf`

## Sample Schema

```
create table messages (
    id      serial primary key,
    subject varchar(4092),
    content text,
    author  varchar(255)
);
```

## Basic Data Types and Functions

### ◆ Data types

- tsvector
- tsquery

### ◆ Functions

- to\_tsvector
- to\_tsquery
- plainto\_tsquery

## Query Syntax

| plainto_tsquery      | to_tsquery                |
|----------------------|---------------------------|
| full text search     | full & text & search      |
|                      | full & text   search      |
| full & text & search | full & !text   search     |
|                      | (! full   text ) & search |

## The Match Operator @@

- ◆ tsvector @@ tsquery
- ◆ tsquery @@ tsvector
- ◆ text @@ tsquery
  - to\_tsvector(text) @@ tsquery
- ◆ text @@ text
  - to\_tsvector(text) @@ plainto\_tsquery(text)

*Note that there is no tsquery @@ text.*

## Query Examples

- ◆ Find the messages that contain “computer programs” in the content
- ◆ Find the messages that contain “computer programs” in either the content or the subject

## Create an Index on Text Column(s)

```
create index messages_content_index
  on messages
  using gin(to_tsvector('english',content));
```

- ◆ Expression (function) index
- ◆ The *language* parameter is required in both index construction and query

## Use a Separate Column for Text Search

- ❖ Create a tsvector column
- ❖ Use a trigger to update the column

## Create an Index on the tsvector Column

```
create index messages_tsv_index  
on messages  
using gin(tsv);
```

- ❖ The *language* parameter is no longer required

## More Functions

- ❖ `setweight(tsvector, "char")`
  - A: 1.0
  - B: 0.4
  - C: 0.2
  - D: 0.1
- ❖ `ts_rank(tsvector, tsquery)`
- ❖ `ts_headline(text, tsquery)`

## Function Examples

- ❖ Set the weight of *subject* to be "A" and the weight of *content* to be "D"
- ❖ List the results by their relevancy scores and highlight the query terms in the results

## Using Native SQL in JPA

```
String sql = "select * from employees where id = ?";  
entityManager.createNativeQuery(sql, Employee.class)  
.setParameter(1, employeeId)  
.getResultList();
```

## Named Query in Entity Class

```
@Entity  
@Table( name="employees" )  
@NamedQueries({  
    @NamedQuery( name="employee.findAll",  
        query="select * from employees" ),  
    @NamedQuery( name="employee.findById",  
        query="from Employee where id = :id" )  
})  
public class Employee { .... }
```

A named query can be JPQL or SQL.

## Named Query in Hibernate Mapping File

```
<sql-query name="message.search">
    <return class="Message" />
    <![CDATA[
        select * from messages
        where tsv @@ plainto_tsquery(?)
    ]]>
</sql-query>
```

## Using Named Query in DAO

```
entityManager
    .createNamedQuery("employee.findAll", Employee.class)
    .getResultList();

entityManager
    .createNamedQuery("employee.findById", Employee.class)
    .setParameter("id", employeeId)
    .getSingleResult();
```

## Example: Course Search in CSNS2

- ◆ Course
- ◆ CourseDao and CourseDaoImpl
- ◆ NamedQueries.hbm.xml
- ◆ csns-create.sql

## FTS in Databases vs. Standalone Libraries

- ◆ Pros??
- ◆ Cons??