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".
  
  ```sql
  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 (documents)

Characteristics of FTS

- Vs. Databases
  - Relevancy ranking
  - "Fuzzy" query processing
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

- Stripping non-textual data
- Tokenizing
- Removing stop words
- Stemming
- Indexing

Document

- Original
  
  ```html
  <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:
  - [the] [solution] [sshd] [login] [errors]

Problem of stop word removal??
Stemming

◆ Reduce a word to its stem or root form.
◆ Examples:
  - connection, connections → connect
  - connected, connecting
  - connective
  - solution, sshd, login, errors → solve, sshd, login, error

Inverted Index

Query Processing

Query → tokenizing → Removing stop words → Stemming → Searching → results → Ranking

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/2_3_2/api/core/overview-summary.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

Document

A document consists of a number of user-defined fields

- Title: FTS with Lucene
- Author: Chengyu Sun
- Content:
lots of words ...
lots of words ...

Types of Fields

- Indexed – whether the field is indexed
  - Tokenized
  - Untokenized
- Stored – whether the original text is stored together with the index

Common Usage of Field Types

<table>
<thead>
<tr>
<th>Field</th>
<th>Tokenized</th>
<th>Indexed</th>
<th>Stored</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Large text file</td>
<td>Y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ID, people's name, date</td>
<td>Y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non-searchable data</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
</tbody>
</table>
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()

Example 2: Search

- Query and QueryParser
- IndexSearcher
- Hits
- 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 )
- close()

Hits

- A ranked list of documents used to hold
  search results
- Methods
  - Document doc( int n )
  - int id( int n )
  - int length()
  - float score( int n ) – normalized score
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 name )
  - Field getField( String name )

Handle Rich Text Documents

- HTML
  - NekoHTML, JTidy, TagSoup
- PDF
  - PDFBox
- MS Word
  - TextMining, POI

Example: FTS in Evelyn

- Indexer and Searcher interface
- FileHandler interface
- File handler implementations
  - DefaultFileHandler
  - TextFileHandler
  - HtmlFileHandler
  -PdfFileHandler
- Spring beans configuration

Further Readings

- Lucene in Action by Otis Gospodnetic and Erik Hatcher

FTS in PostgreSQL

- Since 8.3
  - tsearch/tsearch2 module before 8.3
Sample Schema

```sql
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

```
plaintext to_tsquery
full text search
  full & text & search
  full & text | search
  full & !text | search
  (! full | text ) & search
```

The Match Operator @@

```
tsvector @@ tsquery
tquery @@ tsvector
text @@ tsquery
  to_tsvector(text) @@ tsquery
  text @@ text
  to_tsvector(text) @@ plainto_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)

```sql
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

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

- The \textit{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 Hibernate

- Example:

  ```sql
  SQLQuery query = session.createSQLQuery("select * from messages");
  query.addEntity(Message.class);
  List messages = query.list();
  ```

Named Query ...

In Hibernate mapping file:

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

```java
public List searchMessages(String query)
{
    return getHibernateTemplate()
        .findByNamedQuery("message.search", query);
}
```

Example in CSNS

- Search mailing list messages
  - Message.java
  - Message.hbm.xml
  - MessageDao.java
  - MessageDaoImpl.java

FTS in Databases vs. Standalone Libraries

- Pros??
- Cons??