ΗΥ-464:
Συστήματα Ανάκτησης Πληροφορίας
Information Retrieval Systems

Πανεπιστήμιο Κρήτης, Άνοιξη 2006

Φροντιστήριο 3

Θέμα : Open Source Information Retrieval Systems

Ημερομηνία : 23 Μαρτίου 2006

Outline

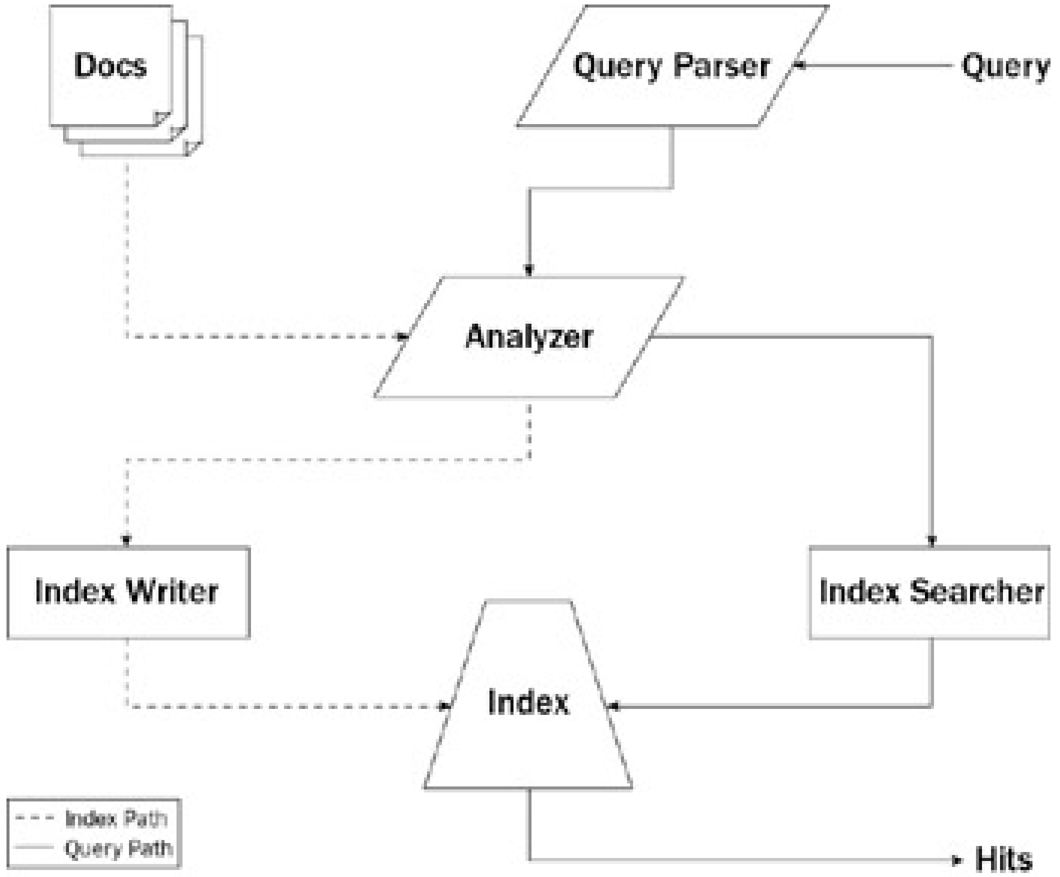
- Lucene
  - What is Lucene?
  - Search Engine Paths
  - Document Feeds
  - Lucene API
- Google
  - Web Services
  - API

Introduction

- Open-source **API** maintained by Apache Software Foundation's Jakarta Project.
- Implemented in JAVA
  - Ported in C++, .NET , Perl and Python.
- Incredibly **MODULAR**, the developer can use it however he wants.
- What makes a Search Engine really tick?
  - it's ***effectiveness***.
- How do we measure effectiveness:
  - **Accuracy :** percentage of documents that were returned from the available set of documents for a particular query.
  - **Precision :** percentage of documents returned that are actually about the particular query.

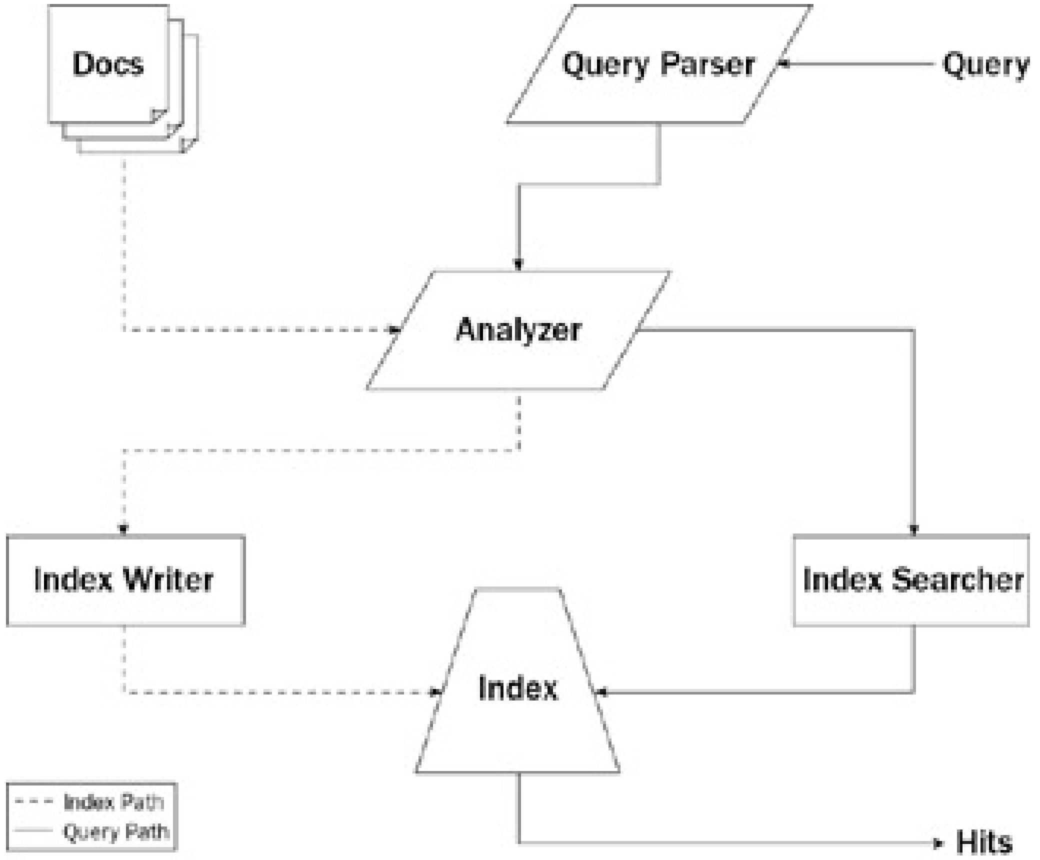
Search Engine Paths (1/2)

- Document Feeding
  - The analyzer turns docs to weighted terms



Search Engine Paths (2/2)

- ***IndexSearcher*** performs the search of the index
- Indexes rarely ever hold the entire document, a set of **Hits** are returned, where each hit represents what is retained about the document within the index.



Indexing strategies

“*When the user starts the application for the first time, Scishare will use the default settings that identifies the user called PSEUDO USER. Pseudo users are provided with automatically generated X.509 certificates and have access to public resources.*”

Removing stop words ..

“user starts application first time, Scishare use default settings identifies user called PSEUDO USER. Pseudo users provided automatically generated X.509 certificates access public resources.”

Stemming..

*“user **start** application first time, Scishare use default **setting identifies** user called PSEUDO USER. Pseudo **users provided automatically generated** X.509 **certificates** access public **resources**.”*

Document Feeding

- Support for various documents formats
  - Power Point
  - Portable Document Format
  - Xml / xsl
  - Microsoft Word Document
  - And more…

Lucene API (1/2)

- To **index** documents, you write a method that performs the following steps:
  - Set a list of files
  - Create a Document instance for each file
  - Create an instance of **Analyzer**. This could be the included **StandardAnalyzer** or one as sophisticated as you can make it.
  - Create an **IndexWriter** with the following: a location of where to locate the index, an instance of the Analyzer just created, and a flag to tell it whether to create the index or not (if it is missing).
  - Add the Document objects to the **IndexWriter** using the **addDocument** method.

Lucene API (2/2)

Imports: StandardAnalyzer, IndexWriter, FileDocument;

```
private void indexFiles()
{…
    IndexWriter writer = new IndexWriter("lib/indexInfo", new            StandardAnalyzer(), true);
    indexDocs(writer,new File("lib/parseDoc"));
    writer.optimize();
    writer.close(); …
}
public static void indexDocs(IndexWriter writer, File file)throws Exception
{
    if (file.isDirectory())
    {
            String[] files = file.list();
            for (int i = 0; i < files.length; i++)
                    indexDocs(writer, new File(file, files[i]));
    }
    else
            writer.addDocument(FileDocument.Document(file));
}
```

Query (1/2)

- Create a **QueryParser** by passing in the default field to search (as a String) and an instance of **Analyzer**.
- Call parse on **QueryParser** to return a **Query** object.
- Initialize an **IndexSearcher** with the location of the index you wish to search.
- Pass the Query object into the search method of **IndexSearcher**, which will return a **Hits** object where the Hits object is a ranked list of the Document objects.

Query (2/2)

```
try
{
    File lib = new File("lib");
    Analyzer analyzer = new StandardAnalyzer();
    Searcher searcher = new IndexSearcher(lib.getCanonicalPath() + "/indexInfo");
    Query query = QueryParser.parse(searchStr, "contents", analyzer);
    Hits hits = searcher.search(query);
    …
    Document doc = hits.doc(i);
    path = doc.get("path");
    …
    searcher.close();

}
```

QueryParser

- Can parse queries such as:
  - **free AND "text search“**
  - **+text search**
  - **giants –football**
  - **author:pillai java**
- Lucene also lets you write your own Analyzer to accommodate the sophistication desired for a particular application.

Applications

- Searchable email.
- Searchable WebPages.
- Website search.
- Content search.
- Version control and content management.
- Source code searching

Criteria

- Supported search mechanisms
  - Algorithms
  - Data structures
- Document feeding
- Interfacing and API
- Extensions

Questions

References

- http://lucene.apache.org/java/docs/api/index.html
- http://cnlp.org/presentations/slides/AdvancedLucene.pdf
- http://lucene.apache.org/