NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA, SURATHKAL

Department of Computer Science and Engineering

INTEGRATED DESKTOP SEARCH ENGINE

Distributed Computing Systems
Course Project Report

Submitted to

Mr. Annappa

Submitted by

Nitesh Sisodiya (09CO64)

Sharath HP (09CO87)

Date of Submission: 20th April 2012

TABLE OF CONTENTS

1. ABSTRACT

1.1 UNDERSTANDING AND ANALYSIS

- There are various desktop applications used for searching files located in the local system. However one might get the motivation to find a solution to making this facility in a distributed scenario so that we can get documents dispersed in various locations.

- Ex - In a company environment, an employee can search for a particular document, say a leave application form, which he might easily obtain through the distributed desktop search engine.

- Using a desktop search tool can be very efficient with high performance even in presence of huge amounts of data, as large as terabytes!

- How is it done - This is achieved by maintaining an index database of all these files. Moreover it is more important and relevant today where a company stores a myriad of different documents in various operating systems distributed across many desktops.

- On every system we need to have a UPnP server and controller. During the UPnP discovery phase, the requester who wants to search for files acts as the UPnP controller and the rest act as UPnP servers providing the desktop search service. When a user wants to do desktop search, he will send the desktop search request to all the servers and receive responses from all of them.

- In the paper they are discussing the implementation of an OS independent, search engine independent procedure to share desktop search results in the intranet. There UPnP controllers and servers run on each machine which interact with each other.

- What they are using - A UPnP ‘Search’ action in XML format is specified so that the UPnP servers can extract the relevant information from them based on the arguments and parameters specified in it.

- Why they use it - This ‘action’ is processed by the servers and appropriate parameters passed on to the actual search engine to do the search. The results of the search engine are then again packed in XML format again and returns it as UPnP action argument named ‘results’ which are interpreted by the controllers and subsequently displayed at the requester side.

- Also in the XML schema of the search ‘result’ certain other parameters can be encapsulated. This could include a unique ‘result id’.

- What is the use - The requester could then use to make a download request for that particular file. The server can then make a copy the file to a mini HTTP server and return the URL for the requester to download. With this mechanism certain security issues may be addressed. For instance the search engine may look for the file location and the final URL may be generated only if access to this file is permitted.

1.2 IMPLEMENTATION

Upon understanding the description given in the paper we plan to implement a simple GUI based Integrated Desktop Search Engine for the windows platform. It is an abridged version of the one described in the paper as we will not be using open XML format to communicate between peers and instead rely only on raw socket functions to achieve the same (passing parameters as information without packing it in XML).

We will try to build a distributed application that enables remote search (as well as local) for files in the intranet.

For the purpose of implementing the search engine we will be using lucene core and some additional support packages to support the development of the application.

We will be using java as the language of implementation. Making socket calls between peers is easy and communication is main part of our distributed application and creating threads and handling them is efficient, so it is fit that we use java for the same. Apart from this one more motivation is the java swing framework which provides a very nice interface for building the GUI of the software.

2. INTRODUCTION

Usability of intranet can be changed tangibly if an effective search tool is used. A good search engine ensures that :-

- Users find what they're looking for, first time, regardless of the format or location of the information.
- A wide variety of information can be effectively dispersed and made available to staff, without the need for complex navigation systems or filing conventions.
- It is cost-effective and expands to suit growing requirements can be a much better.

It is important to recognize that every intranet is different, with its own objectives, requirements and environment. Most intranets evolve over time, and search functionality need not be a daunting task. A search tool can be implemented quickly, and then refined as the intranet grows and the needs of the organization change.

The desktop search is a tool that uses certain keywords to search various data sources like i) the web browser histories, ii) the E-Mail archives, iii) the text documents or iv) the metadata of mp3 files from the local disk storage. After the search tool is installed, desktop search engine used by the search tool will parse all files stored on local disks and maintain an index database for achieving reasonable performance especially when the local storages have kept data of several hundred gigabytes or even terabytes. **The key techniques** used by the desktop search engine are the ability to parse all files in various formats and the ability to do full text search to the parsed contents.

It is convenient to find information that is hidden in the myriad files in the local hard drives with the help with desktop search tool.

Following are some of the most important features of our application:

- The application is DISTRIBUTED

  Documents to be searched may not reside on your local machine. Being distributed it can make queries to other machines in the network.

- The application uses P2P sharing

  No special requirements for server just install and get connected. Each peer serves others.

- It is FAST

For fast search it creates indexes of the shared folders. Creation of indexes considerably reduces the search time for a query.

3. IMPLEMENTATION DETAILS

We have used **Lucene** Package (API) for indexing the content of files and subsequent searching.

3.1 APACHE LUCENE

Apache Lucene (TM) is a high-performance, full-featured text search engine library written entirely in Java. It is a technology suitable for nearly any application that requires full-text search, especially cross-platform. Lucene offers powerful features through a simple API

Features

Scalable, High-Performance Indexing

- over 95GB/hour on modern hardware
- small RAM requirements -- only 1MB heap
- incremental indexing as fast as batch indexing
- index size roughly 20-30% the size of text indexed

Powerful, Accurate and Efficient Search Algorithms

- ranked searching -- best results returned first
- many powerful query types: phrase queries, wildcard queries, proximity queries, range queries and more
- fielded searching (e.g., title, author, contents)
- date-range searching
- sorting by any field
- multiple-index searching with merged results
- allows simultaneous update and searching

3.2 FUNCTIONAL COMPOSITION:

- **Indexing Files**: During the indexing step, a document class is prepared from the files to be indexed from a particular directory and it is added to the index.
- **Searching for Files:** This is the process of consulting the search index and retrieving the documents matching the Query (based on name or content).
- **Render Results**: Once we have the raw set of documents that match the query, sorted in the right order, we then render them to the user in an intuitive, consumable manner.

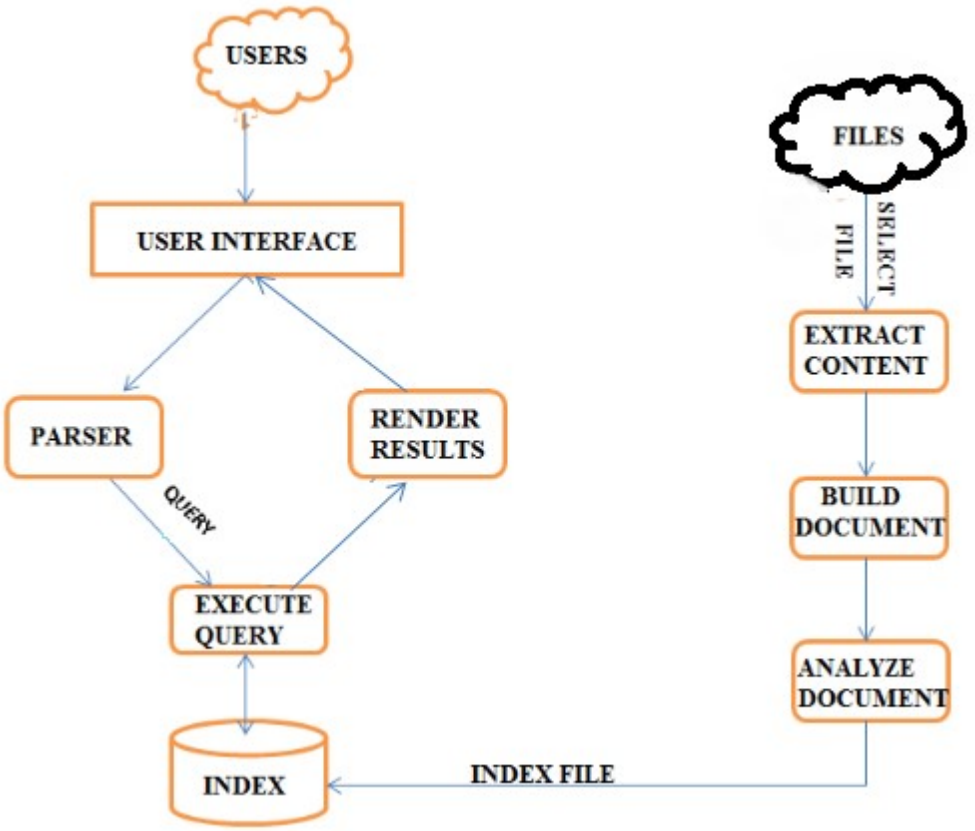

Fig: 2.1 Typical components view of search application

3.3 CORE INDEXING CLASSES USED

In our Indexer class, we need the following classes to perform the indexing procedure:

1. **IndexWriter:** IndexWriter is the central component of the indexing process. This class creates a new index or opens an existing one, and then adds, removes or updates documents in the index.

2. **Analyzer:** Before text is indexed, it's passed through an Analyzer. The Analyzer, specified in the IndexWriter constructor, is in charge of extracting those tokens out of text that should be indexed, and eliminating the rest.

3. **Document:** A Document represents a collection of fields. It can be like virtual document—a chunk of data, such as a web page, an email message, or a text files—that you want to make retrievable at a later time. Fields of a document represent the document or meta-data associated with that document.

4. **Field:** Each Document in an index contains one or more named fields, embodied in a class called Field. Each field has a name and corresponding value, and a bunch of options, described in Section , that control precisely how Lucene will index the Field's value.

5. **Directory:** The Directory class represents the location of a Lucene index. It's an abstract class that allows its subclasses to store the index as they see fit. In our Indexer example, we created an FSDirectory, which stores real files in a directory in the filesystem, and passed it to IndexWriter's constructor.

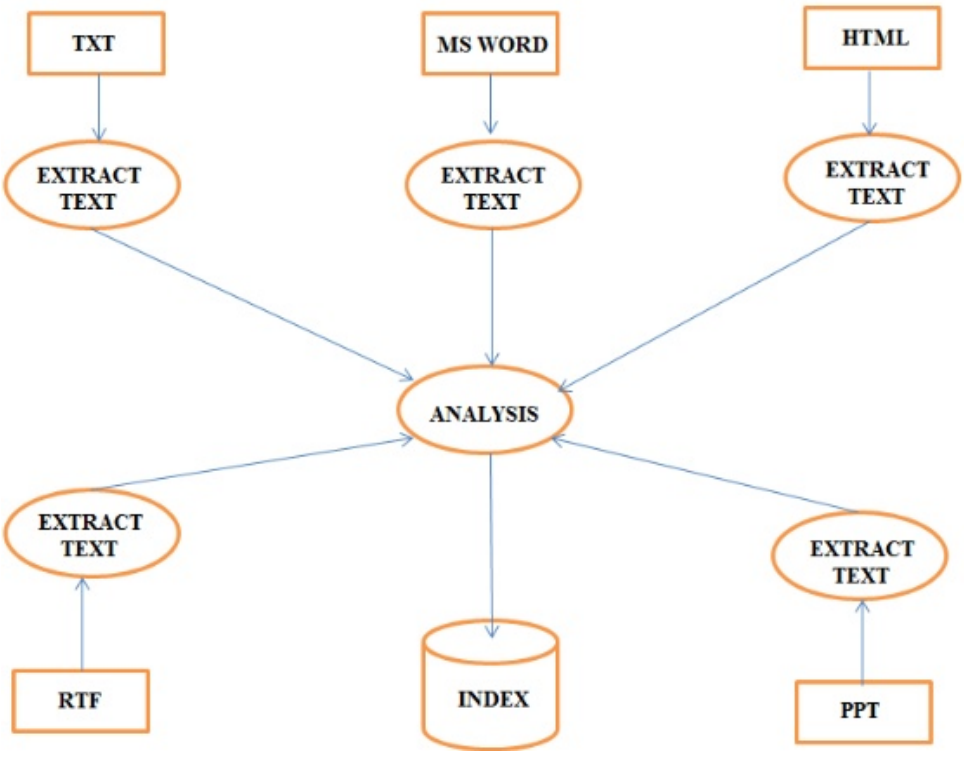

Fig 3.2: The process of indexing

3.4 CORE SEARCHING CLASSES USED

The basic search interface that Lucene provides is as straightforward as the one for indexing. Only a few classes are needed to perform the basic search operation:

1. **Query:** Lucene comes with a number of concrete Query subclasses. The most basic Lucene Query: TermQuery. Other Query types are BooleanQuery, PhraseQuery, PrefixQuery, PhrasePrefixQuery, RangeQuery, FilteredQuery, and SpanQuery.

2. **Term:** A Term is the basic unit for searching. Similar to the Field object, it consists of a pair of string elements: the name of the field and the word (text value) of that field.

3. **IndexSearcher:** IndexSearcher is to searching what IndexWriter is to indexing: the central link to the index that exposes several search methods.

4. **TermQuery:** TermQuery is the most basic type of query supported by Lucene, and it's one of the primitive query types. It's used for matching documents that contain fields with specific values.

5. **TopDocs:** The TopDocs class is a simple container of pointers to the top N ranked search results—documents that match a given query

During indexing, the text is first extracted from the original content and used to create an instance of Document; containing Field instances hold the content. The text in the fields is then analyzed, to produce a stream of tokens. Finally, those tokens are added to the index in a segmented architecture.

3.5 FTP PROTOCOL

- File Transfer Protocol (FTP) is a standard network protocol used to copy a file from one host to another over a TCP/IP-based network, such as the Internet. FTP is built on client-server architecture and utilizes separate control and data connections between the client and server applications which solve the problem of different end host configurations (i.e. Operating System, file names). FTP is used with user-based password authentication or with anonymous user access.

- FTP itself uses the TCP transport protocol exclusively, or in other words, it never uses UDP for its transport needs. Typically an application layer protocol will use one or the other. One notable exception to that is DNS or Domain Name System. FTP also is odd in the fact that it uses two ports to accomplish its task. It typically uses port 20 for data transfer and port 21 to listen to commands. Though having data transferred over port 20 is not always the case as it can also be a different port as well. That is where the confusing part for many people comes into play. There are two modes to FTP, namely active and passive mode. These two modes are initiated by the FTP client, and then acted upon by the FTP server.

- The query entered will be fired simultaneously on both local machine and remote machine and the results are displayed obtained from both local and remote machine. By selecting a particular file we can retrieve a file from the remote machine using FTP Protocol.

4. APPLICATION CORE

4.1 APPLICATION START

There are totally three components that are executed once our application starts. One is the server thread (provider of search service), second is the fileserver thread (provider of download service) and third the application's main window (the controller in our case).

We use a custom built class called 'AppView' in our implementation. It takes care of the GUI and button click events triggering request for service (search) and index functions.

4.2 SERVER

The server thread is responsible for accepting remote requests and processing them. It does so by launching threads for request from each peer. The thread that is created then handles further processing. This way we can establish concurrency.

We have built a class called 'Server' for this purpose. An object of this class is instantiated (in AppView) every time you run the application. It extends the thread class hence executes independent of the main class.

4.3 FILESERVER

The fileserver thread is responsible for accepting file download requests from the peers. The fileserver in-turn launches threads for handling these requests. If the file is found it is transferred to the peer through basic socket function calls. If not found returns a 'File Not Found' reply.

We have built the 'Server1' class that provides this service. This is similar to the Server thread except provides a different service (download).

4.4 APPLICATION RUN

- With the help of the application the users index the directories that they want to share with the peers. Once the index is ready, the application can successfully return results of queries that match the files details stored in the index directory.

- A user can make a search request and this triggers the application to search the local as well as the specified remote machines. The search request, as mentioned earlier is handled by the server running on the respective hosts.
- All requested hosts reply to the requesting peer (both successful and unsuccessful replies). The results are then displayed in a text area on the user interface at the peer side.
- The result consists of the host address and the corresponding file location on that host. Using these parameters the user can then choose to download the file by making a request to that host. The request is sent by the application and the fileserver running on the host receives it.
- Upon verification at the host (checking if file - corresponding to the path entered at the user side - exists) the file is then transferred to the user using basic socket functions.
- In the application the user has the option to choose from a list of file types to index or search. Also within the search he can specify whether to search for the file-name or the file-contents.

5. APPLICATION SCREENSHOTS

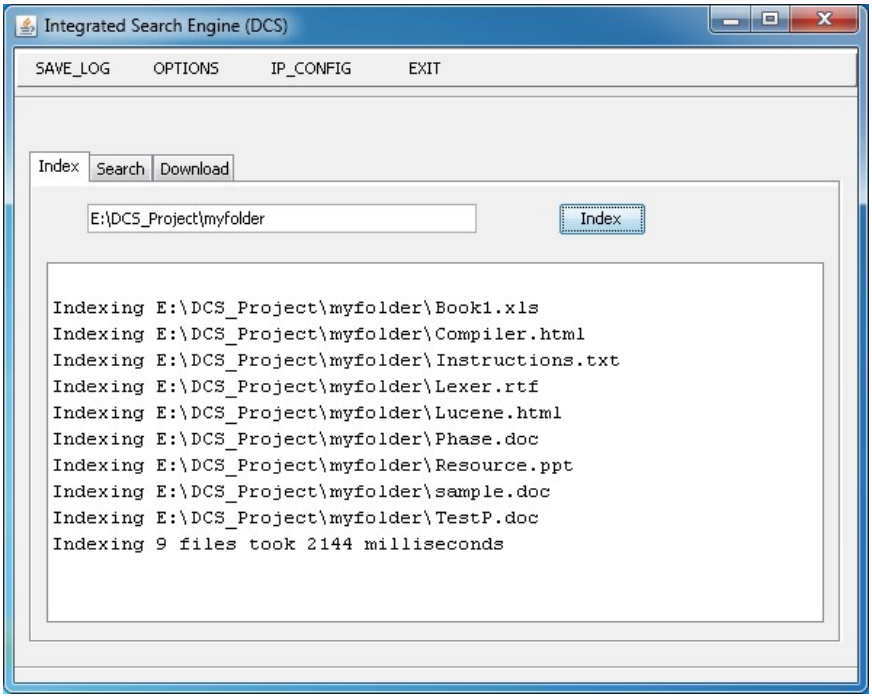

Fig 5.1: Indexing Files

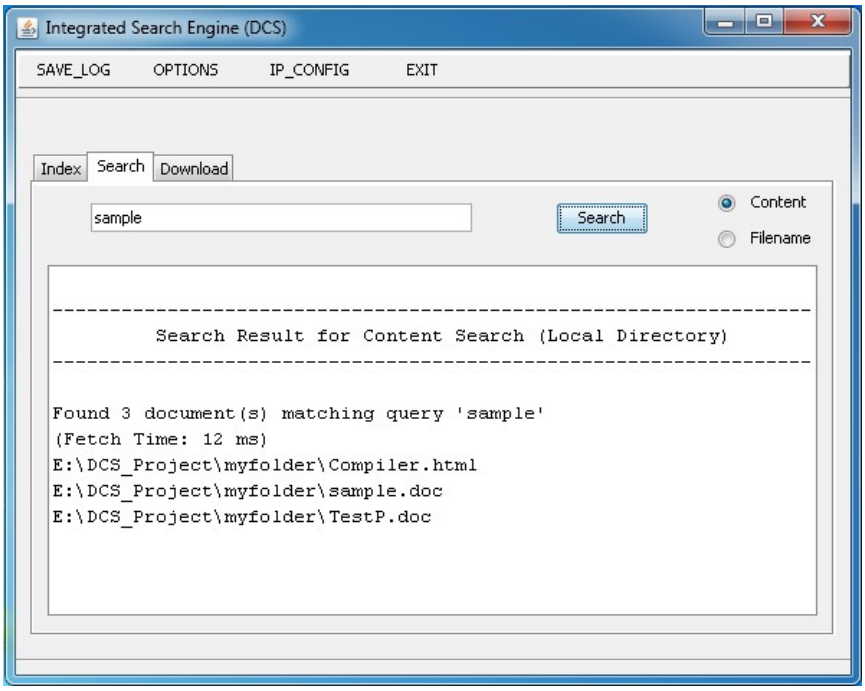

Fig 5.2: Searching Files

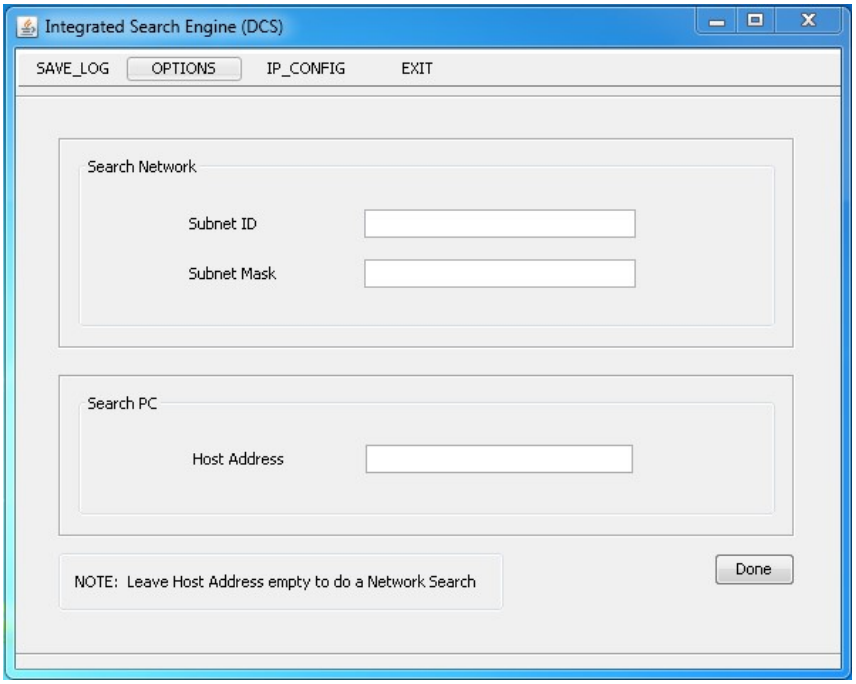

Fig 5.3: Search Configuration

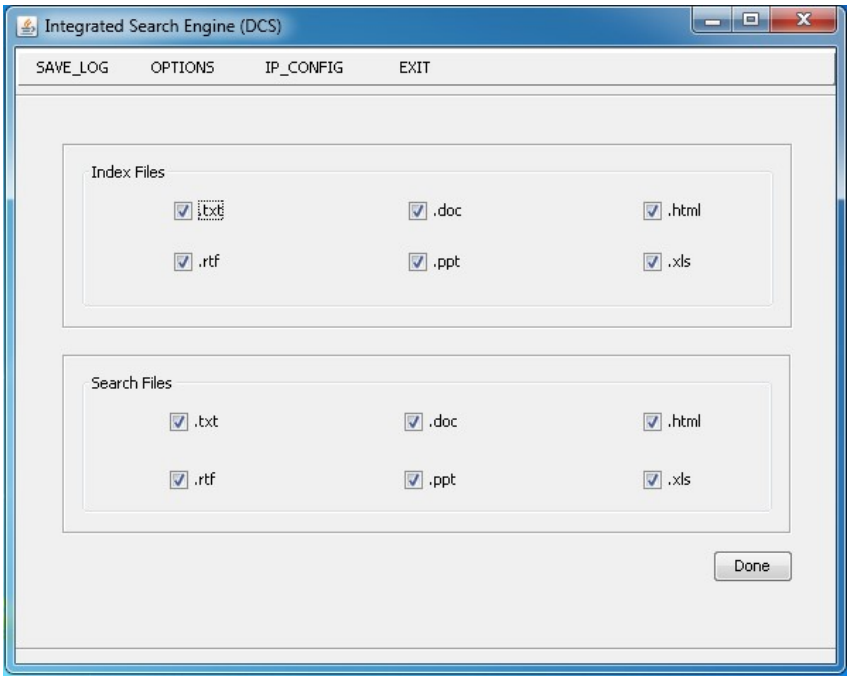

Fig 5.4: Select File Types

6. FUTURE WORK

- The implementation is a abridged verison of that described in the IEEE paper. In that the distributed application communicates with its instances on the peers through open XML format, which may use SOAP messages that are being passed. But our version of the distributed app uses basic socket calls to communicate with each other. We could also make use of this open format to make it more flexible and possibly make interaction with other applications possible.

- For transfering files between peers we make use of basic socket calls to keep the implementation simple. We could instead use RMI (Remote Method Invocation) in Java for fast transfer of files between peers.

- Also we can incorporate several other file types such as pdf, xml files etc… These require special parsers to convert them into extractable text format. This is because lucene requires files in one or other extractable text format.

7. REFERENCES

[1] Wei Lun Huang,Tzao Lin Lee, Chiao Szu Liao,Dept. of Comput. Sci. & Inf. Eng., Nat. Taiwan Univ, Taipei,"Desktop search in the intranet with integrated desktop search engines"Computer Systems Architecture Conference, 2008. ACSAC 2008. 13th Asia-Pacific Aug. 2008 page(s): 1 - 4

[2] Computerweekly web site, http://www.computerweekly.com/Articles/2006/04/25/215622/security-special-report-who-sees-your-data.htm, Apr. 2008.

[3] Google desktop web site, http://desktop.google.com, March. 2010

[4] Beagle desktop search engine web site, http://beagle-project.org/Main_Page, Apr. 2010

[5] www.lucene.apache.org

[6] http://today.java.net

[7] www.lucenetutorial.com

[8] http://darksleep.com/lucene/