Class

CSC9010 - Search Technologies and Personal Information Management

Instructor

Dr. Lillian Cassel

Semester

Fall 2008

Team 3

- Andres Narvaez
- Charlie Han
- Scott Weaver
- Sowmya Moturi
- Tarun Kothuri

Project

siQuest - related documents finder

Introduction

siQuest is a Web application that lets users to upload multiple files and find related documents on the Internet using popular search engines, such as Google, Yahoo, and Microsoft Live. The application is based on the siQuest framework, which includes modules to extract texts from various file formats, index the documents, and provide API (Application Programming Interface) for extracting the indexed data. Built on top of the framework is the Query Builder Engine, which uses the information retrieved from the framework to construct a set of useful terms to be used as the query string by various search engine APIs. The framework and the query builder engine provides the core of the application. Finally, the siQuest core, or siQuest Engine, is wrapped by the UI and Web service framework, where the UI framework provides human interaction and the Web service provides an interface to other computing frameworks via SOAP and REST. The open source search engine library Apache Lucene/JAVA was used to aid with the searching operations.

Text Extraction

Lucene can only index a specific document format. Because of this, we must convert from arbitrary file formats to the document format if we want to index different file formats. While various Lucene document converters do exist, it was easier to create our own documents using text extraction libraries. SiQuest currently supports txt, pdf, html, doc, ppt, xls, xml, and docx file formats. Text support is essentially built into Lucene, so implementation of it was trivial. PDF and HTML text extraction were both done using a text manipulation library called Multivalent. The three old Microsoft Office formats, Word, PowerPoint, and Excel, are done using the Apache POI library, which contains many functions for text extraction in these formats. XML is done using a Java XML library. The Microsoft Word 2007 format, docx, was the most difficult, because it first required the usage of a Java unzip library, and then the proper internal XML file had to be parsed for the text.

The Framework - Indexer, Analyzer

The Lucene API provides the building blocks to both index and search documents. The purpose of the SiQuest Framework is to take those smaller blocks and build the functionality that is needed by the SiQuest application. The following functionality is available through the SqIndexer class:

Creating a blank index, adding documents to the index, adding an entire directory of documents to the index, and deleting an index. In order to index a document the SqTextExtractor and SqDocumentFactory classes are needed. The SqTextExtractor extracts the text based on the document type while the SqDocumentFactory creates a document that can be indexed by Lucene.

The purpose of the SqAnalyzerFactory class is to specify to Lucene how to parse the text. The framework provides five different analyzers which are white space, simple, stop, standard, and snowball. The white space analyzer simply parses the text into terms using white space as the delimiter. The simple analyzer builds on the white space analyzer by converting the terms to lowercase. The stop analyzer builds on the simple analyzer by excluding words in a stop list. The standard analyzer also builds on the simple analyzer by cleaning up punctuation and having the option to use stop words. The snowball analyzer enhances the standard analyzer by stemming the terms.

Another class included in the framework is the SqInfoExtractor class which provides the following functionality: retrieving the number documents that exist in the index, retrieving the names of the fields that are being indexed, retrieving all the terms, retrieving the high frequency terms, verifying the existence of a document, retrieving the id of the document, and retrieving a document and all it's information. Although SqSearcher class isn't used by the SiQuest application the functionality exists for searching for a document and analyzing documents without inserting it into the index. The final class in the framework is the SqPreferences class which is used throughout the application to store and retrieve preferences for the application. Together these classes form and easy to use API that the rest of the SiQuest application can use.

Query Building

The query builder takes information about the indexed documents from the framework, and returns the list of terms that ought to be used by the search engine APIs. Search API specific features are not implemented as part of this project release (except a limitation of up to 19 keywords for the Microsoft Live Search engine, since it would choke when 20 or more keywords are passed as input).

In order to generate a query string with most relevant keyword terms that would result in best results, a scoring scheme is used to rank the keywords within the index. The *score* is calculated as *score* = (sum-*tf* / *idf*) * 100

where

sum-*tf* is the sum of the normalized term frequency (tf) for a particular keyword *k* in a document *d*, over all documents
idf is the inverted document frequency of the term *k* over all indexed documents

The term sum-*tf* gives an estimate of how important (frequently occurring) the specific keyword is among the documents, whereas the term *idf* gives an idea of how unique a particular keyword is among the documents.

idf would be multiplied in case of searching; it would score a document highly with the uniqueness factor of a given keyword.
The query builder needs the exact opposite characteristics; the score should be higher when a keyword is more common among the documents. Therefore, the term *idf* is yet again inverted by dividing it from the sum of term frequency (sum-*tf*) of the keyword. The term 100 is multiplied for easier manipulation and comparisons.

The query builder returns the terms that scored higher than the 98-th percentile of the terms. If the number of terms returned a too few or too many, depending on user preferences settings, the number of terms will be adjusted from the final result.

Web Service - Computer Interface

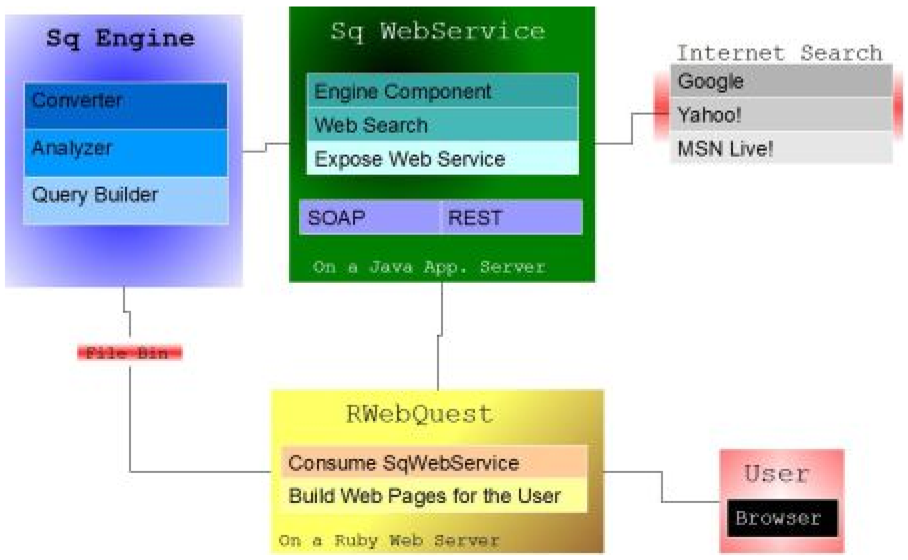
The siQuest Engine generates queries appropriate to the document, but this query should be used to perform a search, and the functionality of the service should be exposed or made available so that it can be used by a search application.
The siQuest WebService system does the above. It contains three major components

1. An Engine Component which performs the document analyzing and query building.
2. A Web Search component which makes queries to Google, Yahoo! And Live! Search engines through their SOAP or REST Search APIs using the query generated by the engine and collates the results.
3. A Web Service interface to which other applications can connect and utilize the siQuest functionality.

The Sq WebService is a Java Web Application based on the Servlet 2.5 Spec, and the SOAP Web Service endpoints are enabled by the Apache Axis framework. This service can be run on any web application server the likes of Apache Tomcat. A WSDL File was created containing all the available methods and their messages. Any application which wishes to use siQuest's SOAP API can locate the wsdl at http://server/siQuestService/wsdl/siQuestSOAPSOAP.wsdl (where server is the host name of the machine where this web service is hosted) and perform requests using SOAP's XML based messages.

A much simpler REST interface is also provided, where requests can be made by simple http request containing the filenames(multiple files are separated by a colon) and analyzer name as parameters. The response containing the search results is provided in JSON (Java Script Object Notation) format. A sample request is "http://server/siQuestService/SiQuestREST?files=filenames&analyzer=analyzername"

A Web based client called RWebQuest was also written for the above service. This client consumes the services provided by SqWebService using its REST API. This website runs on the Ruby on Rails framework which is an agile framework for rapid development of web applications. This site provides capability for selecting/uploading files and making requests to the SqWebService and showing the results. This was done to demonstrate the flexibility of web services, in which the web service engine is written in Java and is being consumed by an application written in Ruby.



A visual representation of the structure of siQuest with focus on the WebService, The engine components are also detailed and complex.

The UI - Human Interface

The project has a simple web user interface.It allows user to select/upload the documents, multiple documents can be uploaded at a time. The User has an option of choosing from the different Analyzers and modify results accordingly. The combined results received are displayed as a list of links and their description. A Tag cloud of the query terms is also generated. The cloud is displayed along with the search results in the same page. The tag cloud algorithm makes use of the most important keywords identified from the previous stages. The Weight of each word is calculated from the index score and an appropriate style is chosen. To represent different words in different font sizes CSS styling is used, 10 different styles are used each matching the weight of the keyword.

Summary

The overall process is user uploads file(s) and the client makes a request to the siQuest WebService which invokes the siQuest Engine where the document(s) are indexed, analyzer, appropriate search query generated which is used by the siQuest WebService to search on Google, Yahoo! and MSN Live! search engines, the results are collated and returned to the client which displays the results as desired. The entire system was found to be very effective in finding pages related to the given documents.

References

- Apache Lucene - http://lucene.apache.org/java/docs/
- Multivalent - http://multivalent.sourceforge.net/
- Apache POI - http://poi.apache.org/
- Apache Axis SOAP Engine - http://ws.apache.org/axis/
- Google AJAX Search API - http://code.google.com/apis/ajaxsearch/
- Yahoo! Search API - http://developer.yahoo.com/search/
- Microsoft Live! Search API - http://dev.live.com/livesearch/
- Ruby on Rails - http://www.rubyonrails.org/
- Open Source Web Design - http://www.oswd.org/