# Advanced Information Retrieval Web Services for Digital Libraries

Weimao Ke, Yueyu Fu and Javed Mostafa
Laboratory of Applied Informatics Research
Indiana University, Bloomington
IN, 47405-3907
(812)856-4182, 01
{wke, yufu, jm}@indiana.edu

## ABSTRACT
Web service, as a standardized XML-based protocol, has been useful for inter-system communication and integration. However, web services in the IR domain have not been widely used. In a previous paper [2], we discussed our implementation of an information retrieval (IR) function called LUCAS, a web service for extracting, weighing, and ranking terms. It properly demonstrated adaptability and accessibility of web services in IR. In this paper, we are going to discuss a more advanced version developed recently called Lucas II. This updated implementation includes functions of term generation, clustering, and document classification that can be operated in different knowledge domains.

## Categories and Subject Descriptors
H.3.5 [**Information Storage and Retrieval**]: Online Information Services – *web-based services.*

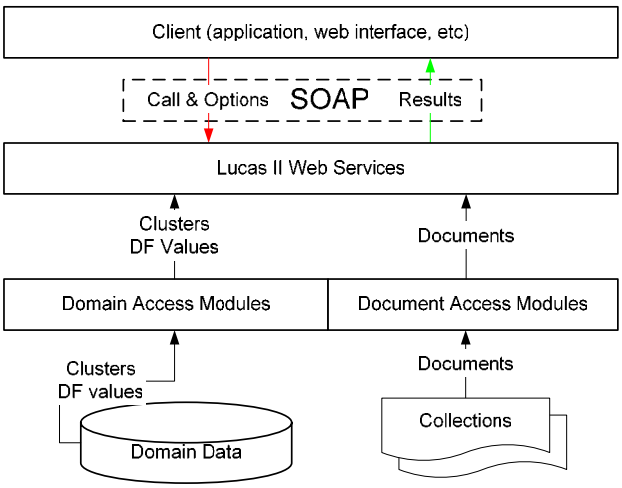
## General Terms
Design, Experimentation

## Keywords
Web Services, Information Retrieval

## 1. INTRODUCTION
In this paper, we concentrate on three important IR functions (term generation, cluster generation, and document classification) and demonstrate how these can be offered as web services for supporting basic digital library functions. The paper will briefly discuss the importance of the IR functions and outline the architecture of our system. It will describe primary parameters of the web services and give examples of accessing them in multiple ways. The paper will conclude with a discussion of future work.

## 2. IMPORTANCE OF THE FUNCTIONS
The web services we offered are fundamental functions of IR. They can be applied to a variety of digital library practices. For instance, term generation can be used for indexing, information extraction, and summarization while clustering is useful in searching and indexing. In addition, classification is important as related to indexing and filtering. Web services encapsulating these functions are widely applicable to projects in this area.

## 3. ARCHITECTURE
The system (see Figure 1) mainly consists of three components: the Lucas II web services, which are deployed on Tomcat and Apache Axis server; the client, which passes the user selected parameters to the web services and gets the results back based on the SOAP protocol; and data access modules to access domain terms and document collections. We refer to this system as Library of User-Oriented Concepts for Access Services II (i.e. LUCAS II).




* Arrow: direction of information flow

**Figure 1. System Architecture**

## 4. WEB SERVICES
In Lucas II, we developed and deployed three web service methods (operations). Each of them is detailed as below.

**I. Term generation**: The term generation function retrieves terms from a given online document (URL), computes the term weights, and sorts the result:

1) Extract all the terms except the stop words from a given document and count term frequency.
2) Retrieve the DF values from the domain DB for all the extracted terms and compute TF*IDF weights.
3) Sort the terms based on the TF*IDF values and select a number of the top terms specified by the user.

The method for term generation is ***generateTerms*** with the following parameters: 1) ***domain***: the knowledge domain with term DF values; 2) ***URL***: the URL of a web document; 3) ***numberTerms***: the number of top terms requested; 4) ***showWeights***: showing the weights of terms or not; 5) ***format***: format of the result [0 Text | 1 HTML]. The service returns a list of the extracted terms.

**II. Cluster Generation:** This function has two steps: term extraction and term clustering. It is based on an algorithm developed by Mostafa, Quiroga and Palakal [3], which employs a word distribution weighing scheme and some heuristics to extract terms. The whole process is defined below:

1) Extract common terms that are among the top weighted terms throughout a certain number of documents in the collection.
2) Compute term-term associations based on the extracted list of terms and the doc-term matrix of the collection.
3) Apply a distance threshold and cluster the terms with centroids.

The method for cluster generation is ***generateClusters*** with parameters as follow: 1) ***domain***: the knowledge domain; 2) ***R***: consider the top R ranked tokens in each document; 3) ***D***: the percentage of documents that must contain a token ranked above R for the token to be selected; 4) ***theta***: vectors must be theta far away from all existing centroids to be considered a new centroid; 5) ***numberClusters***: number of clusters requested. There are other parameters for formatting the result. Lucas II returns a list of clusters and their member terms.

**III. Document Classification**: This web service classifies an online document (URL) into a proper cluster after computing its similarity scores with a set of term clusters:

1) Convert the list of term clusters into cluster-term vectors.
2) Retrieve the document content and use the unique terms in the clusters to render its doc-term vector (binary, frequency, or TF*IDF representation).
3) Compute the similarity score between the document and each of the clusters using Dice or Cosine algorithm.
4) Sort the clusters based on similarity scores and choose the top cluster.

The method for document classification is ***classifyURL*** with parameters as follow: 1) ***domain***: the knowledge domain where DF values can be obtained; 2) ***docURL***: the URL of a document; 3) ***clusterString***: a list of term clusters that can also be generated through the cluster generation web service; 3) ***repAlgorithm***: representation model [0 Binary | 1 Term Frequency | 2 TF-IDF]; 4) ***classAlgorithm***: Classification algorithm [0 Dice | 1 Cosine]. The service returns the best-matched cluster and similarity score of the document to each cluster.

For more information about Lucas II web services, please refer to http://tara.slis.indiana.edu:8080/lucas2/lucas2.html.

## 5. WEB SERVICE CLIENTS

There are multiple ways to invoke these web services. Based on our web service description and the SOAP protocol, new client interfaces can be easily built according to users' preferences. One way is to use a Java application. A sample Java code for the cluster generation service can be downloaded at: http://tara.slis.indiana.edu:8080/lucas2/LucasClient2.java.

Another way is to use JSP/Servlet to enable accessing these web services on any web browser. As shown in Figure 2, a user can select options through the web interface and submit (post) the requests to a JSP/Servlet component, which then communicates with the "classifyURL" web service and transfers the result back to the browser. This demo JSP client can be accessed at: http://tara.slis.indiana.edu:8080/lucas2/lucas2class.jsp.

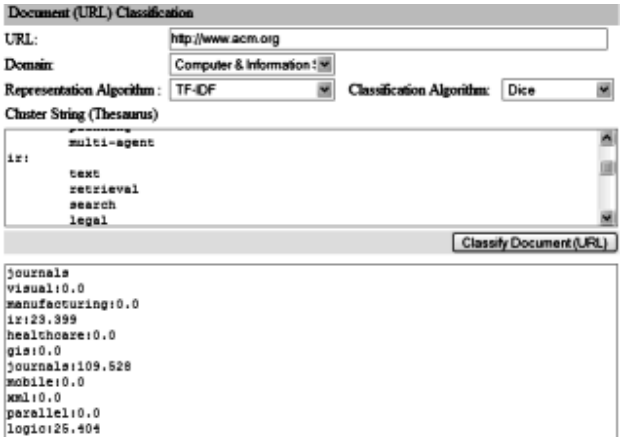



**Figure 2. A JSP/Servlet client for document classification**

## 6. CONCLUSION

Our system implementation has demonstrated that IR algorithms can be effectively turned into web services, which can be accessed in a variety of ways. This flexibility enables easier integration of IR systems and/or algorithms without duplicate efforts. Future implementations of LUCAS may be to integrate existing web services for more sophisticated IR functionality. Such web services will have great potential for supporting digital library operations. In fact, the term generation and classification services have been successfully integrated into one of our digital library projects called ENABLE to generate index terms and classify web pages automatically. For more information about the ENABLE project, please visit: http://enable.slis.indiana.edu.

## 7. ACKNOWLEDGMENTS

This work was partially supported through a grant from the National Science Foundation Award#:0333623.

## 8. REFERENCES

[1] Chen, H. Semantic Research for Digital Libraries. *D-Lib Magazine*, 5(10). 1999.

[2] Fu, Y., and Mostafa, J. "Toward Information Retrieval Web Services for Digital Libraries". *IEEE/ACM Joint Conference on Digital Libraries 2004*, Tucson, Arizona, 2004.

[3] Mostafa, J., Quiroga, L., and Palakal, M. Filtering Medical Documents Using Automated and Human Classification Methods. *Journal of the American Society for Information Science*, 49(14), 1998.

[4] Truner, M., Budgen, D., and Brereton, P. Turning Software into a Service. *IEEE Computer*, 36(10), 2003.