

# The BINGO! Focused Crawler: From Bookmarks to Archetypes

Sergej Sizov, Stefan Siersdorfer, Martin Theobald, Gerhard Weikum  
University of the Saarland, Saarbruecken, Germany  
WWW: <http://www-dbs.cs.uni-sb.de>  
E-Mail: {sizov, weikum}@cs.uni-sb.de

## 1. Motivation

Focused crawling is a relatively new, promising approach to improving the recall of expert search on the Web. Consider an advanced Web user, say a researcher or a student, who is looking for the mathematical definition of Chernoff bounds or for the text (and possibly the chords) of the Western song "Raw Hide". With today's best Web search engines this search is likely to end up with frustration. Typically, search engines provide some potentially relevant documents and the user could find better results within the neighbourhood of these sites (viewing the Web as a graph), but manually surfing hundreds or thousands of Web pages is out of the question for time and cost reasons (i.e., the cost of the "intellectual cycles" spent by the human user). Often the best results can be obtained from portals like [www.yahoo.com](http://www.yahoo.com) or [www.invisibleweb.com](http://www.invisibleweb.com) where documents are intellectually preclassified into a hierarchy of topics, also known as an ontology. Unfortunately, maintaining such an ontology with human experts (or cheap students) as classifiers is barely feasible in the long term. This is where focused crawling [CBD99a, CBD99b] kicks in: it starts from a user- or community-specific tree of topics along with a few training documents for each tree node, and then crawls the Web with focus on these topics of interest. This process can either build a personalized, hierarchical ontology whose tree nodes are populated with relevant high-quality documents, or it can be initiated to process a single expert query such as the ones above (i.e., viewing the query terms as an initial training document).

According to [CBD99a] the key components of a focused crawler are a document classifier to test whether a visited document fits into one of the specified topics of interest, and a distiller to identify the best URLs for the crawl frontier (i.e., those hyperlinks in already visited documents that, when traversed, promise the best results in the continuation of the crawl). Obviously the distiller should be aware of the specified topics, too, to keep the crawl on focus. So for both components the quality of the training data is the most critical issue and potential bottleneck for the effectivity and scale of a focused

crawler. The BINGO!<sup>1</sup> system implements an approach to focused crawling that aims to overcome the limitations of the initial training data. To this end, BINGO! identifies, among the crawled and positively classified documents of a topic, characteristic "archetypes" and uses them for periodically re-training the classifier; this way the crawler is dynamically adapted based on the most significant documents seen so far. Two kinds of archetypes are considered: good authorities as determined by employing Kleinberg's link analysis algorithm, and documents that have been automatically classified with high confidence using a linear SVM classifier.

## 2. System Overview

The BINGO! system consists of six main components that are depicted in Figure 1: the crawler itself, an HTML document analyzer that produces a feature vector for each document, the SVM classifier with its training data, the feature selection as a "noise-reduction" filter for the classifier, the link analysis module for determining topic-specific authorities and hubs, and the training module for the classifier that is invoked for periodic re-training. BINGO! is implemented as a collection of Java servlets that run under the control of a Web application server. All intermediate data, such as "raw documents" that the crawler fetches or feature vectors (vectors of term weights), and also the final ontology index are stored in a database using Oracle8i. This way the various components and their execution threads are largely decoupled with regard to timing and performance.

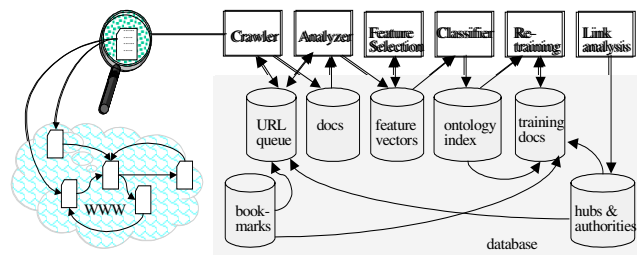


Fig. 1: The BINGO! architecture and its main components

<sup>1</sup> BINGO! stands for bookmark-induced gathering of information.

### 3. System Components

The crawler starts from a user's bookmark file that serves two purposes: 1) it provides the initial seeds for the crawl (i.e., documents whose outgoing hyperlinks are traversed by the crawler), and 2) it provides the initial contents for the user's hierarchical ontology and the initial training data for the classifier. Bookmark files are organized into a hierarchy of folders as illustrated by the example of Figure 2. The names of these folders are treated as the topics of the focused crawler, and the entire hierarchy forms an ontology tree whose nodes are the topics and whose edges represent subtopic relationships. The documents associated with each node are interpreted as characteristic training data for the corresponding topic:

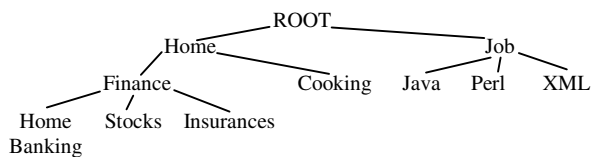


Fig. 2: User bookmarks for initialization of the ontology

Each crawled and analyzed document is handed over to the classifier. BINGO! uses an open-source implementation [Bio] of a support-vector-machine (SVM) classifier [Vap98, Bur98]. The classifier has been trained by the initial training data on a per topic basis; so for each node of the ontology tree a mathematical decision model has been built, with control parameters derived from the node's training data, that determines whether a new document belongs to the topic with high confidence or not.

The classification of a new document proceeds in a top-down manner starting from the root of the ontology tree. For each node the classifier returns a yes-or-no decision and a confidence measure of the decision. The document is assigned to the node with the highest confidence in a yes decision. Then the classification proceeds with the children of this node, until eventually a leaf node is reached.

The large dimensionality of initial feature vectors would create efficiency problems for the classifier. Furthermore, the unrestricted feature vectors contain way too many "noisy" terms that would lead the classifier astray. The BINGO! engine provides advanced techniques for stopword elimination, stemming using the Porter algorithm [BR99], and feature selection based on the Mutual Information criterion (also known as cross-entropy or Kullback-Leibler divergence) [MS99].

BINGO! periodically initiates re-training of the classifier, whenever a certain number of documents have been crawled or successfully classified. At such points, a new set of training documents is determined for each node of the ontology tree. For this purpose, a set of the most characteristic documents of a topic, coined

archetypes, are determined in two, complementary, ways. First, a link analysis procedure, using Kleinberg's HITS algorithm [Kl99], is initiated. This procedure yields a ranking of authorities for each topic, documents that contain high quality information relevant to the topic. As a side effect, the link analysis also computes the best hubs for a topic, good sources of links to authorities; these hubs are given high priority in the crawler's URL queue. The second source of topic-specific archetypes builds on the confidence of the classifier's yes-or-no decision for a given node of the ontology tree. Among the automatically classified documents of a topic those documents whose yes decision had the highest confidence measure are selected as archetypes.

BINGO! is implemented completely in Java, with some components implemented as stored procedures under the Java virtual machine of Oracle8i and other components as Java servlets under the Apache web server. Further information about BINGO! can be found at the URL <http://www-dbs.cs.uni-sb.de/~bingo>

### 4. About the Demo

The demo will show the BINGO! focused crawler operating on a medium-sized collection of HTML documents that are preloaded onto a notebook and crawled via a local proxy. Bookmarks for a small personalized ontology serve as seeds for the crawl and as initial training data for the SVM classifier. The demo will highlight the influence of feature selection and periodic re-training based on the novel concept of archetypes. It will be seen that the accuracy of the ontology's documents (i.e., the fraction of the documents that truly fit into the corresponding topic) improves substantially as the crawl proceeds and leverages the periodic re-training.

### References

- [Bio] The Open-source BioJava Project <http://www.biojava.org>
- [BR99] R. Baeza-Yates, B. Ribeiro-Neto: Modern Information Retrieval, Addison Wesley, 1999.
- [Bur98] C.J.C. Burges: A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, Vol.2, No.2, 1998.
- [CBD99a] S. Chakrabarti, M. van den Berg, B. Dom: Focused Crawling: A New Approach to Topic-specific Web Resource Discovery, WWW Conference, Toronto, 1999.
- [CBD99b] S. Chakrabarti, M. van den Berg, B. Dom: Distributed Hypertext Resource Discovery through Examples, VLDB Conference, Edinburgh, 1999.
- [Kl99] J.M. Kleinberg: Authoritative Sources in a Hyperlinked Environment, Journal of the ACM, Vol. 46, No. 5, 1999.
- [MS99] C.D. Manning, H. Schuetze: Foundations of Statistical Natural Language Processing, MIT Press, 1999.
- [Vap98] V. Vapnik: Statistical Learning Theory. Wiley, New York, 1998.