From Focused Crawling to Expert Information: an Application Framework for Web Exploration and Portal Generation

Sergej Sizov, Jens Graupmann, Martin Theobald

University of the Saarland
Department of Computer Science
P.O. Box 151150, 66041 Saarbruecken, Germany
{sizov,graupman,mtheobald}@cs.uni-sb.de

1 Introduction

Focused crawling is a relatively new, promising approach to improving the recall of expert search on the Web. It typically 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 efficiently build a theme-specific, hierarchical directory whose nodes are populated with relevant high-quality documents for expert Web search.

The BINGO! focused crawler implements an approach 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 (good authorities as determined by Kleinberg's HITS algorithm, and documents classified with high confidence using a linear SVM) and uses them for periodically retraining the classifier; this way the crawler is dynamically adapted based on the most significant documents seen so far.

While a large amount of information can be collected from the "Surface Web" with traditional crawling as done by today's popular search engines, the major part of high quality, topic-specific data is stored in searchable databases that only produce results dynamically in response to a direct request (i.e., the "Hidden Web" or "Deep Web"). Automated meta portal generation for these hidden sources comes with all the traditional problems a meta search engine has to face.

Proceedings of the 29th VLDB Conference, Berlin, Germany, 2003

The demonstration shows our approach towards fully automated portal generation that merely starts with a small set of user-specific training documents and dynamically builds up a unified database of Surface Web data as well as of indexed Deep Web pages derived from on-the-fly generated Web Service interfaces for form pages leveraging Semantic-Web-style ontologies. The prototype platform has been used for generating two applications that illustrate the effectiveness and versatility of our approach: the Handicrafts Information Portal (HIP) built for the Saarland's Chamber of Trades and Small Businesses, and a movie metaportal coined MIPS.

In the following sections we give a short overview of the BINGO! prototype system and then outline the above mentioned application demos.

2 Focused Crawling with BINGO!

The framework uses the BINGO! focused crawler developed by our group [18, 19]. BINGO! has initially been built for crawling Surface Web pages covering HTML, XML, and PDF files; it is now being extended by several components for automated Deep Web search. To this end, we collect pages with HTML forms and consider a subset of these pages as portals; then we post queries to these pages and automatically classify the returned result pages to extend our thematic directory. BINGO! itself consists of several components sketched in the following subsections.

2.1 Feature Space Construction

Our engine uses words as the basic representational units of the document, the standard bag-of-words document model with tf-idf-based feature weights [1]. However, the large dimensionality of initial full-length vectors would create efficiency problems for the classifier. Furthermore, the unrestricted feature vectors contain too many "noisy" terms that would lead the

classifier astray. The BINGO! engine includes techniques for stopword elimination, stemming, and feature selection based on the Mutual Information criterion (also known as relative entropy or Kullback-Leibler divergence) [2, 1].

2.2 Document Classification

The classifier is an exchangeable component that can be registered at run-time. We are currently using linear Support Vector Machines (SVM) as classification method [6, 5, 8]. The SVM method is a learning algorithm that consists of a supervised training phase for each topic, where a separating hyperplane defined as $\vec{w} \cdot \vec{x} + b = 0$ is computed such that it maximizes the margin between a set of positive and negative feature vectors in the m-dimensional feature space. The classification step for a test document $\vec{y}$ then simply consists of computing the sign of a decision function of the form $\vec{w} \cdot \vec{y} + b > 0$, where the sign denotes on which side of the hyperplane the test vector is determined to be. The BINGO! engine uses Thorsten Joachim's SVMLight [4] for this purpose.

2.3 Link Analysis

The link structure between documents in each topic is an additional source of information about how well they capture the topic. The BINGO! engine applies Kleinberg's link analysis method, coined HITS [3], to compute authority scores for the documents of each topic in the directory tree.

2.4 Retraining

Our recent efforts have focused on a *semi-supervised retraining step* [18, 19] based on the automated detection of topic-specific "archetypes" that are found in the hyperlink neighborhood of the initial training samples and positively classified into the given topic with high confidence. BINGO! uses archetypes for periodically retraining 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*, where confidence is derived from the distance of a test document to the separating hyperplane of the linear SVM [6, 5].

A separate SVM is learned for each node of the given directory tree. The *hierarchical* classification for the entire tree is implemented by recursively assigning a test document to the node among a set of siblings for which the positive classification confidence is highest [6].

2.5 Overcoming Portal Borders

The Deep Web comprises all information that resides in autonomous databases behind portals, and this data cannot be reached by traditional crawlers (unless a portal offers an explicit link collection in addition to the usual query form). For proceeding beyond these portals the focused crawler automatically generates query data for the input parameters of a portal's HTML forms. To avoid portal-specific mappings between a user query and each portal's forms, a portal is automatically encapsulated as Web Service whose functionality can be described in the generic WSDL format.

Portal wrapping: When an initial crawl discovers a portal candidate, a component is invoked that applies heuristic rules for generating a WSDL entry on the fly. Typically, highlighted text next to form fields will become parameter names, and the type of a form field determines the corresponding parameter type (e.g., an enumeration type for fields with a pull-down menu) [17]. The WSDL entry (and also a UDDI entry) can either be registered automatically, or can be placed in a candidate queue for later inspection and possible modification by a human.

Ontology searching: The parameter names of a WSDL interface are viewed as characteristic semantic concepts and inserted into a locally maintained ontology index. Furthermore, for enumeration types the possible values are likely to be in close semantic relationship to the corresponding concept [14] and are also added to the ontology index. We query our internal ontology service (the WordNet [12, 13] ontology with edges enriched by similarity weights) for newly found concepts and values, extract related words and their relationships, and add them to the ontology index, too. The external ontologies are manually registered with our service, and are themselves wrapped as Web Service.

Portal query generation: The focused crawl is driven by either a topic description, given in the form of training data, or an explicit user query. In either case it attempts to explore the data behind portals by automatically generating queries to the portals and indexing the returned result pages. The query generator attempts to match keywords from the training data or concept names from the user query with parameter names of the portal [9, 10, 11]. If this fails, the local ontology index is searched for related terms and the matching is reattempted. Once appropriate parameters have been identified for portal search, a limited number of parameter values are generated from the values in the query or topic-specific keywords, with preference given to words that have an is-instance-of semantic relationship to the parameter name [15, 16].

We implemented a framework that automatically generates wrapper components for portals. The wrappers encapsulate the communication with the corre-

sponding Web portal and enable the crawler to systematically explore the source's "hidden" content.

3 Demonstration

3.1 Handicrafts Information Portal (HIP)

In this application, the focused crawler BINGO! was used to produce the search engine for the HIP portal (**H**andicrafts **I**nformation **P**ortal) of the Saarland's Chamber of Trades and Small Businesses. The HIP portal has been designed to meet special information demands of small handicraft businesses such as special laws and regulations, financial subsidies, environmental regulations and programs, information for trainees, etc. A typical information that, for example, an electrician might ask from the portal is: are there any new EU (European Union) regulations regarding the proper disposal of electronic parts (computer boards, TV sets, etc.).

For simplified navigation and search (a baker and a computer scientist would expect completely different results for the query "protocol bakery"), the portal provides orthogonal topic hierarchies for three basic groups of information demands:

- professions and professional groups of the branch (electrician, plumber, auto mechanic, etc.)
- typical processes and workflows (accounting and billing, maintenance of health and safety standards, etc.)
- education and careers of handicrafts (trainee, apprentice, master, etc.)

Each hierarchy contains currently 3 levels with a total of 15 to 25 categories. To focus the crawler on these themes, we used online tutorials for handicrafts, topic-specific laws and regulations, as well as homepages of companies from appropriate businesses. Each category was initially populated with 10 to 15 manually selected training documents.

In the learning phase, the crawl was restricted to the hosts of the initial sources and to a depth of 3. After re-training on the original training data and the automatically selected archetypes, the focused crawl was continued on the Web without host, domain, or depth limitations. A typical run of the BINGO! engine collects up to 10.000 positively classified documents within 12 hours and visits about 10 million pages on several 10.000 different hosts. The crawled data is stored in a MySQL database for further processing.

The search interface is implemented as a collection of servlets implemented in the scripting language PHP. The engine provides several advanced functions for expert search:

- It supports different ranking schemes such as SVM confidence measures, authority scores produced by the HITS algorithm, cosine similarity for user queries, user feedback score, last modified attribute (and also combinations of these metrics). The last option is useful to highlight modified Web sources for the "what's new" search.
- It supports advanced search options for a result page selected by the user. For example, it is possible to query the neighborhood (predecessors and successors) of a page or restrict a search to all results from the same host, to find similar documents.
- It supports session-based query refinement on search results.

To improve the quality of the portal's data, HIP provides mechanisms for user feedback: suggestion of new Web sources and topics of interest, reporting classification errors, evaluation of the usefulness of visited search results.

For better manageability by the HIP portal administrator, the BINGO! focused crawler, originally implemented as a stand-alone Java program, was adapted for execution under Apache Tomcat VM in connection with a MySQL database on the same server. The administration interface is implemented as a collection of PHP and JSP pages for particular administration tasks such as crawler parametrization, start/stop and pause/resume of a crawl, maintenance of training sources, evaluation of user feedback, etc. In addition, the administration toolkit allows automated clustering of crawl results for a given topic. For better recognition of potential new topics of interest, the resulting clusters are annotated by cluster-specific terms using Mutual Information as a selection criterion.

Starting from the HIP topic hierarchy (initially populated with few manually pre-selected training sources), the current prototype demonstrates the positive effects of the learning phase with retraining based on automatically selected archetypes and the subsequent harvesting phase with rich crawling results. The demo is concluded by the presentation of the resulting portal interface with advanced functions for expert querying. The final version of the search engine (with more comprehensive volume of searchable data and improved performance) will be available for public use by the end of 2003.

3.2 The MIPS Movie Portal

As a second application (for mere demo purposes) we generated a movie metaportal that we coined MIPS (Movie Information Portal and Search Assistant) This portal integrates static HTML pages as well as Deep Web sources such as IMDB and Amazon which have been discovered in an initial focused crawl for movie pages and selected by their authority and classification confidence scores [19]. Part of the crawl analysis is the detection of form fields in the retrieved pages.

When potentially relevant forms are found, the Web Service Generator is invoked. The Web Service Generator automatically creates a wrapper for the form, thus providing a query interface in WSDL. The system then attempts to classify the Web Service into *movie genres*, based on a list of available genres and small sets of representative movies as training data. On the basis of this information the system generates queries for each Web Service and each genre. When the Web Service returns results that fit with the training data of the inquired genre, the portal is added to the corresponding topic. If the Web Service does not qualify for any genre, it is removed from the database.

The arguments of the MIPS search form are mapped to the WSDL parameters of a Web Service. For this purpose, the system has its own ontology modelling synonym and hypernym relationships for the movie domain. For performance purposes this mapping is precomputed in advance and stored in the underlying BINGO! database. For a given query, the query processor first retrieves the best matches among the indexed static HTML pages. Then it queries the UDDI registry for matching Web Services and invokes them with the precomputed parameter mapping and the appropriate values. Although the returned pages are built dynamically, their URL usually includes all information to access the page from its source server. So dynamic pages can now be classified and added to the topic directory along with their URLs. For subsequent queries the pages can be searched directly in our index without calling the Web Service again.

The demo shows the operational meta portal about movies that integrates static HTML sources as well as other portals. The GUI allows the user to specify whether she only wants to query the static content or wants to include also the dynamically extracted content of other Web portals. We also show how a portal administrator can easily add new Deep Web sources to the meta portal.

References

[1] R. Baeza-Yates, B. Ribeiro-Neto: Modern Information Retrieval. Addison Wesley, 1999.

[2] C.D. Manning, H. Schuetze: Foundations of Statistical Natural Language Processing. MIT Press, 1999.

[3] J.M. Kleinberg: Authoritative Sources in a Hyperlinked Environment. Journal of the ACM, Vol.46, 1999.

[4] T. Joachims: Learning to Classify Text using Support Vector Machines, Kluwer, 2002.

[5] C.J.C. Burges: A Tutorial on Support Vector Machines for Pattern Recognition. Data Mining and Knowledge Discovery 2(2), 1998.

[6] V. Vapnik: Statistical Learning Theory. Wiley, 1998.

[7] S. Chakrabarti: Mining the Web. Morgan Kaufmann Publishers, 2002.

[8] S. Dumais, H. Chen: Hierarchical Classification of Web Content. ACM SIGIR Conference, 2000.

[9] Special Issue on Organizing and Discovering the Semantic Web, Data Enginieering Bulletin, Vol.25 No.1, 2002.

[10] Special Issue on Integration Management, Data Enginieering Bulletin, Vol.25 No.3, 2002.

[11] J. Madhavan, P.A. Bernstein, P. Domingos, A. Halevy. Representing and reasoning about mappings between domain models. In Eighteenth National Conference on Artificial Intelligence (AAAI), 2002.

[12] C. Fellbaum: WordNet: An Electronic Lexical Database. MIT Press, 1998.

[13] G. Miller: Wordnet: A Lexical Database for English. Communications of the ACM 38(11), 1995. 2000.

[14] A. Maedche, S. Staab: Ontology Learning for the Semantic Web. IEEE Intelligent Systems 16(2), 2001.

[15] G. Panagiotis, L. Gravano, M. Sahami: Probe, Count and Classify: Categorizing Hidden-Web Databases, ACM Sigmod Conference (SIGMOD), 2001.

[16] S. Raghavan, H. Garcia-Molina: Crawling the Hidden Web, 27th International Conference on Very Large Data Bases (VLDB), 2001.

[17] A. Sahuguet, F. Azavant: Building light-weight wrappers for legacy Web data-sources using W4F, 25th Conference on Very Large Data Bases (VLDB), 1999.

[18] S. Sizov, M. Theobald, S. Siersdorfer, G. Weikum: BINGO!: Bookmark-Induced Gathering of Information. Proceedings of the 3rd International Conference on Web Information Systems Engineering (WISE), 2002.

[19] S. Sizov, M. Biwer, J. Graupmann, S. Siersdorfer, M. Theobald, G. Weikum, P. Zimmer: The BINGO! System for Information Portal Generation and Expert Web Search. Proceedings of the First Conference on Innovative Data Systems Research (CIDR), 2003.