Focused Crawler for Discovering the Structure of Web Topics

RICHARD A. WASNIOWSKI
Computer Science Department
California State University
Carson, CA 90807, USA

Abstract: - A web crawler is a system that searches the Web, beginning on a user-designated web page, for web pages that contain a particular target. The web crawler follows all the links on the user-designated web page and all the links on each resulting web page until either there are no more links to follow or it reaches some preset limit on the number of sites searched. A web crawler must keep track of each page that it has searched and each of the pages that contain the target string. Focused crawling is a relatively new, promising approach to improve expert search on the Web. Typically, search engines provide some potentially relevant documents and the user could find better results within the neighborhood of these sites.

Key-Words :- Web searching, crawler, graph

1 Introduction

Today we face so much information on the Web that it is frequently difficult to get what we are looking for. That is why advanced search engines are very useful. Search engines contain databases with information about what is on the Web. Google, the biggest search engine offers the fastest and the easiest way to find information on Internet. Google delivers relevant results to users, usually within less than a fraction of a second. Google is based on advanced search Site Ranking-technique, which makes sure that the most relevant search results always show first. Most of the search engines however, are not specified to a topic. Focused crawling is a relatively new, promising approach to improve searching on the Web. Typically, search engines provide some potentially relevant documents and the user could find better results within the neighborhood 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. This is where focused crawling comes in: it starts from a user specific tree of topics with a few training documents 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 i.e., viewing the query terms as an initial training document. The key components of a focused crawler is a document classifier to test whether a visited document fits into one of the specified topics of interest. There are several methods to create a focused search engine. In all the methods we must use dynamic web pages with a connection to a database. We have developed and implemented an approach to focused crawling , the TORO system [14], that aims to overcome the limitations of the initial training data. Toros are known autonomous communities but the name TORO has been adopted for our Web crawler after our local portal called webToro. TORO identifies, among the crawled classified documents, using a linear SVM classifier[19], of a topic and uses them for periodically re-training the classifier; this way the crawler is dynamically adapted based on the most significant documents.

2 System Overview

The TORO system consists of five main components that are depicted in Figure 1. The crawler itself, extraction and refine modules, performance module and mySQL database. TORO is implemented Java and runs under the control of a Web application server. All intermediate data are stored in a database using database. We wanted our search engine to be free to implement so we wanted a free-to-use database manager ant that is why our database is implemented in mySQL. This way the various components and their execution threads are largely decoupled with regard to timing and performance. The crawler starts from a user's bookmark file that serves two purposes: it provides the initial seeds for the crawl and 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, topics of the focused crawler. Each crawled document is handed over to extraction module. TORO uses an open-source implementation

of a support-vector-machine classifier. The classifier has been trained by the initial training data on a per topic basis; so for each node of the ontology tree a 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.

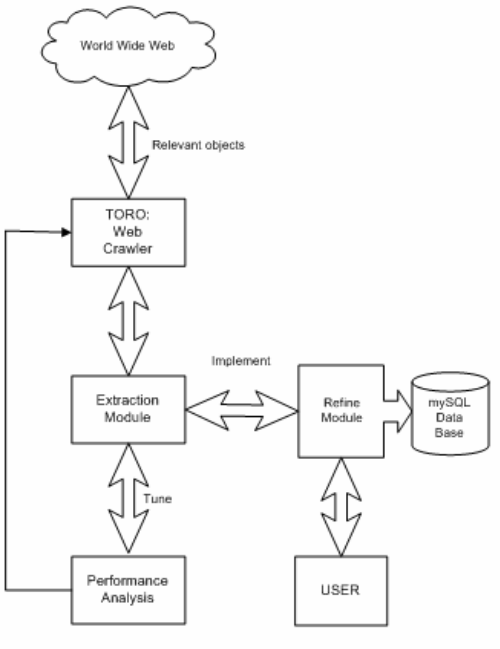

Fig. 1: The TORO architecture and its main components

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. TORO 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. TORO is implemented in Java, with some components implemented as stored procedures under the Java virtual machine and other components under the Apache web server. The main reason why we have chosen to work with Java is because it is platform independent. Java also makes it easy to connect to a MySQL database via JDBC.

3 Experimental results

The TORTO WebCrawler is made for surfing the Web, almost as a user when she visits sites with her web browser. The TORO uses graphical capability similar other web crawlers such as to SPHINX[13].

A good way to measure how well a WebCrawler functions is to check how many pages it can find per time unit. We have done two different analyses that show the speed of our WebCrawler. Our analysis shows that our search engine is almost as good as Google when it comes to finding sites with the queries we have chosen, but gives us better ways of customizing searches. This analysis shows good result even though our TORO WebCrawler is only experimental.

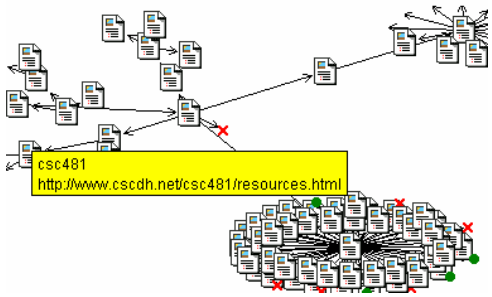

Fig. 2: The TORO uses graphical capability similar other web crawlers such as to SPHINX.

4 Concluding remarks

The Web communities and their boundaries are in a state of continual change and a fascinating source of data. In this paper we have discussed how to built focused crawling - a relatively new, promising approach to improve search on the Web. We have implemented focused web crawler using a set of components including Java and mySQL database. Preliminary results show that our search engine is as good as Google to get relevant pages.

References:

[1] L. A. Adamic and B. A. Huberman. Power-law distribution of the World Wide Web. Science, 287:2115a, 2000.

[2] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. Science, 286:509–512, 1999.

[3] K. Bharat, A. Bröder, M. Henzinger, P. Kumar, and S. Venkatasubramanian. The Connectivity Server: Fast access to linkage information on the Web. In the World Wide Web Conference, Brisbane, Australia, 1998.

[4] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In Proceedings of the 7th World-Wide Web Conference (WWW7), 1998.
[5] S. Chakrabarti, D. A. Gibson, and K. S. McCurley. Surfing the Web backwards. In WWW, volume 8, Toronto, Canada, May 1999. Online at http://www8.org.
[6] B. D. Davison. Topical locality in the Web. In Proceedings of the 23rd Annual International Conference on Research and Development in Information Retrieval (SIGIR 2000), pages 272–279, Athens, Greece, July 2000. ACM.
[7] S. Dill, S. R. Kumar, K. S. McCurley, S. Rajagopalan, D. Sivakumar, and A. Tomkins. Self-similarity in the Web. In VLDB, pages 69–78, Roma, Sept. 2001.
[8] D. D. Lewis. ATTICS: A toolkit for text classification and text mining. In M. Grobelnik, D. Mladenic, and N. Milic-Frayling, editors, KDD-2000 Workshop on Text Mining, Boston, Aug. 2000. SIGKDD, ACM.
[9] F. Menczer. Links tell us about lexical and semantic Web content. Technical Report Computer Science Abstract CS.IR/0108004, arXiv.org, Aug. 2001.
[10] R. Motwani and P. Raghavan. Randomized Algorithms. Cambridge University Press, 1995.
[11] M. Najork and J. Weiner. Breadth-first search crawling yields high-quality pages. In WWW
[12] P. Rusmevichientong, D. M. Pennock, S. Lawrence, and C. L. Giles. Methods for sampling pages uniformly from the World Wide Web. In AAAI Fall Symposium on Using Uncertainty Within Computation, pages 121–128, 2001.
[13] R. C. Miller, K Bharat, SPHINX: a framework for creating personal, site-specific Web crawlers, Computer Networks and ISDN Systems, 30, 1998, 119-130.
[14] R. Wasniowski, TORO: Implementing Focused Crawler Using Java and mySQL, RAW-CS-04-21
[15] S. Chakrabarti, M. van den Berg, B. Dom: Focused Crawling: A New Approach to Topic-specific Web Resource Discovery, WWW Conference, Toronto, 1999.
[16] S. Chakrabarti, M. van den Berg, B. Dom: Distributed Hypertext Resource Discovery through Examples, VLDB Conference, Edinburgh, 1999.
[17] J.M. Kleinberg: Authoritative Sources in a Hyperlinked Environment, Journal of the ACM, Vol. 46, No. 5, 1999.
[18] C.D. Manning, H. Schuetze: Foundations of Statistical Natural Language Processing, MIT Press, 1999.
[19] V. Vapnik: Statistical Learning Theory. Wiley, New York, 1998.