Query- vs. Crawling-based Classification of Searchable Web Databases

Luis Gravano
gravano@cs.columbia.edu
Columbia University

Panagiotis G. Ipeirotis
pirot@cs.columbia.edu
Columbia University

Mehran Sahami
sahami@epiphany.com
E.piphany Inc.

1 Introduction

The World-Wide Web is one of the main channels through which people currently exchange information. Unfortunately, this information is not characterized in a way that would make its semantics readily understandable by computers, which complicates building value-added services on top of the existing information. An ambitious effort that aims to facilitate the development of such services is the so-called "Semantic Web." According to Berners-Lee et al. [1]:

> "The Semantic Web will bring structure to the meaningful content of web pages, creating an environment where software agents roaming from page to page can readily carry out sophisticated tasks for users."

Classification is a useful way to structure web resources. Directories such as *Yahoo!* manually organize pages in a classification scheme that helps users locate information of interest. In principle, each resource could use the Semantic Web infrastructure to categorize itself in a classification scheme of interest. However, since a single, universal taxonomy is unlikely to fit the needs of all users (and applications), a variety of classification schemes will continue to emerge. With a heterogeneity of classification schemes, the self-characterization of resources might prove to be unwieldy. Hence, tools to automatically categorize web resources remain of key importance.

The web contains pages that both regular search-engine crawlers can reach and retrieve in order to index, and documents that search engines ignore. In particular, "hidden-web" databases contain documents that are only accessible through a search interface. Links to documents in a hidden-web database are typically generated only as a response to a dynamically issued query to the database's search interface. As a result, valuable web-accessible content remains largely ignored by current search engines, as the following example illustrates.

Example 1: Consider the medical bibliographic database CANCERLIT® from the National Cancer Institute's International Cancer Information Center, which makes medical bibliographic information about cancer available through the web [1]. If we query CANCERLIT for documents with the keywords `lung AND cancer`, CANCERLIT returns 67,518 matches, corresponding to high-quality citations to medical articles. The abstracts and citations are stored locally at the CANCERLIT site and are not distributed over the web. Unfortunately, the

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

[1] The query interface is available at `http://www.cancer.gov/search/cancer_literature/`.

high-quality contents of CANCERLIT are not "crawlable" by traditional search engines. A query [2] on Google that finds the pages in the CANCERLIT site with the keywords "lung" and "cancer" returns only 186 matches, which do not correspond to CANCERLIT abstracts but rather to other pages in that domain. This illustrates that the valuable content available through CANCERLIT is not indexed by traditional crawlers.

Another scenario in which the contents of a database are not accessible to crawlers is when the database is using a `robots.txt` file to instruct crawlers not to retrieve any files. Technically, in this case a crawler might be able to reach and fetch the pages on the site; however, a crawler implementing proper "netiquette" would not do so. Hence, in this case the database can also be considered uncrawlable. Such is the case with the *PubMed* database [3].

In previous work [8], we proposed a classification method for hidden-web databases. Our technique uses automatically learned queries that are strongly associated with the categories of interest (e.g., the query "lung AND cancer" is associated with the category "Health"). These queries consist of very few keywords, and are adaptively sent to a database that we want to classify. Using only the *number of matches* generated for the queries associated with each category, the algorithm detects the topic distribution in the database and classifies the database accordingly, *without retrieving any documents from the database*. The result of the query probing process is an accurate classification of the database, incurring only a small cost [4].

Our classification method was designed for hidden-web databases, where querying is the only way to access their documents. Interestingly, the same method can be applied to classify any web site that offers a search interface over its pages, whether or not its contents are directly accessible to a crawler. For example, although the news articles at the CNN Sports Illustrated [5] news site are accessible through links, the keyword search interface that is offered can be used alone to classify this site. As an alternative classification approach for such a database, we could use a standard crawler to download all (or a fraction) of the pages at the site, classify the pages using a *document* classifier, and decide how to classify the *site* based on the document class distribution.

The focus of this paper is to compare these two classification approaches, namely the query-based approach that we introduced in [8] and the crawling-based approach outlined above. We present evidence that our query-based approach works best in terms of both classification accuracy and efficiency. In a nutshell, the crawling-based approach can lead to unstable classification decisions, while requiring large amounts of data to be retrieved when classifying large databases. The rest of this paper is organized as follows. Section 2 provides a definition of database classification. Then, Section 3 gives a brief overview of both our query-based algorithm for this task and a crawling-based algorithm. Section 4 reports an experimental comparison of the query- and crawling-based approaches in terms of their accuracy and efficiency. Finally, Section 5 concludes the paper.

2 Classifying Searchable Web Databases

Similar to well-known web directories like *Yahoo!*, other directories *manually* classify hidden-web databases into classification schemes that users can browse to find databases of interest. An example of such a directory is InvisibleWeb [6]. Figure 1 shows a small fraction of InvisibleWeb's classification scheme. The classification of a database into such a scheme is determined by the database contents.

Example 2: Consider topic category *"Basketball." CBS SportsLine* has a large number of articles about basketball and covers not only women's basketball but other basketball leagues as well. It also covers other sports like football, baseball, and hockey. On the other hand, *WNBA* only has articles about women's basketball. The

[2] The query is `lung cancer site:cancer.gov`.

[3] `http://www4.ncbi.nlm.nih.gov/PubMed/`

[4] A prototype implementation is available at `http://qprober.cs.columbia.edu`.

[5] `http://www.cnnsi.com`

[6] `http://www.invisibleweb.com`

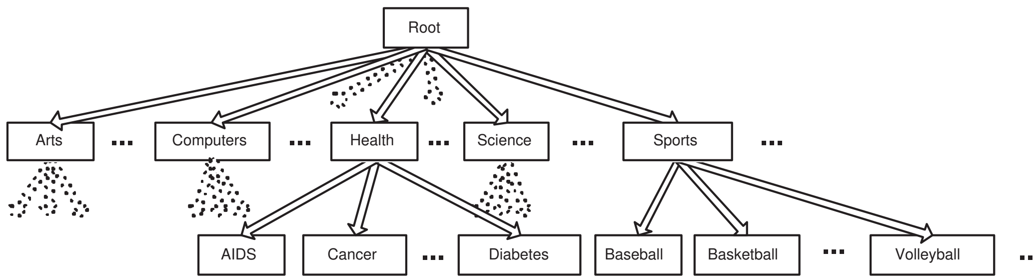

Figure 1: Portion of the InvisibleWeb classification scheme.

way that we will classify these sites depends on the intended usage of the classification. Users who prefer to see *only* articles relevant to basketball might prefer a *specificity-based* classification and would like to have the site *WNBA* classified into node *"Basketball."* However, these users would not want to have *CBS SportsLine* in this node, since this site has a large number of articles not related to basketball.

To define the database categorization problem, we introduce the notion of topic "specificity" of a database. The *Specificity*(D, C_i) of a database D for a given category C_i is the fraction of the documents stored in D that are about C_i [7]. So, a "specificity-based" classification of a database would focus on high-specificity categories. To formalize this notion, we introduce a *specificity threshold* τ_s, with values ranging between 0 and 1. For example, for $\tau_s = 0.25$ we classify a database into a given category only if at least 25% of the database documents are relevant to this category. More generally, we define the classification of a database in a hierarchical classification scheme as follows:

Definition 1: Consider a hierarchical classification scheme C with categories $C_1, \ldots, C_n$, and a searchable web database D. The *ideal classification of D in C* is the set *Ideal(D)* of categories C_i that satisfy the following conditions:

- *Specificity*$(D, C_i) \geq \tau_s$.
- *Specificity*$(D, C_j) \geq \tau_s$ for all ancestors C_j of C_i.
- *Specificity*$(D, C_k) < \tau_s$ for all children C_k of C_i.

where $0 \leq \tau_s \leq 1$ is the given threshold.

Of course, this definition requires a priori knowledge of the distribution of database documents across categories. Unfortunately, this information is rarely available, hence we need a way to estimate it.

3 Query- and Crawling-based Classification Algorithms

We now describe two approaches for classifying searchable web databases, one based on simple queries (Section 3.1) and one based on web crawling (Section 3.2).

[7] In a hierarchical classification scheme, if C_i is not the hierarchy root then this fraction is computed with respect to D's documents in C_i's parent category [8]. Also, please refer to [8] for a complementary notion that focuses only on the absolute number of database documents in each category.

3.1 Query-based Classification

In [8], we presented a query-based algorithm to approximate the *Ideal(D)* categorization of a database D. Our strategy works as follows. First, we train a document classifier over a given classification scheme using machine learning techniques. Second, we construct queries from the document classifier, which we use in turn to probe the database to obtain the category distribution information needed for database categorization. Specifically, we follow the algorithm below:

1. Train a rule-based document classifier with a set of preclassified documents.
2. Transform the learned classifier rules into queries.
3. Adaptively issue these queries to databases, extracting the number of matches for each query.
4. Classify databases using the number of query matches.

Our approach exploits rule-based document classifiers [5] to estimate the number of documents associated with each category. Such a document classifier has rules to assign categories to a document based on the words in the document. For example, the following rules are part of a classifier for the three categories *"Sports," "Health,"* and *"Computers"*:

IF ibm AND computer	THEN Computers
IF jordan AND bulls	THEN Sports
IF cancer	THEN Health

The first rule classifies all documents containing both the words "ibm" and "computer" into the category *"Computers."* Similarly, the third rule classifies all documents containing the word "cancer" into the category *"Health."* Note that all the rules contain conjunctions of words in their antecedents.

To simulate the behavior of a rule-based classifier over all documents of a database, we map each rule of the classifier into a boolean query that is the conjunction of all words in the rule's antecedent. Thus, if we send a query probe to a database, the query will match (a close approximation to) the set of documents in the database that the associated rule would have classified into its corresponding category. For example, we map the rule `IF jordan AND bulls THEN Sports` into the boolean query `jordan AND bulls`. We expect this query to retrieve mostly documents in the *"Sports"* category. Now, instead of retrieving the documents themselves, we just keep the number of matches reported for this query (it is quite common for a database to report the total number of documents matching a query in the results page with a line like "X documents found"), and use this number as a measure of how many documents in the database match the condition of this rule.

The query-probing results provide a good approximation of the document distribution across the categories in a database. Specifically, we can approximate the number of documents in a category as the total number of matches for the query probes associated with the category. Using this information, we can then classify the database according to the specificity threshold τ_s, since we have approximations to all the necessary category-frequency information. For example, in Figure 2 we have listed the detected specificity for the top-level categories of a classification scheme for five searchable web databases. There is a clear peak in the detected specificity for the category where each database would have been classified by a human. Using these numbers, we automatically derive the correct classification of the database. Previous experimental results [8] showed that our method had on average over 75% precision (i.e., on average over 75% of the categories that we produce for a database are correct) and over 75% recall (i.e., on average we discover over 75% of the *Ideal* categories for a database) for a set of 130 real, autonomous web databases. The average number of queries sent to each database was 185, and no documents needed to be retrieved from the databases. Furthermore, the number of words per query ranged between just one and four words. Further details of our algorithm and evaluation are described

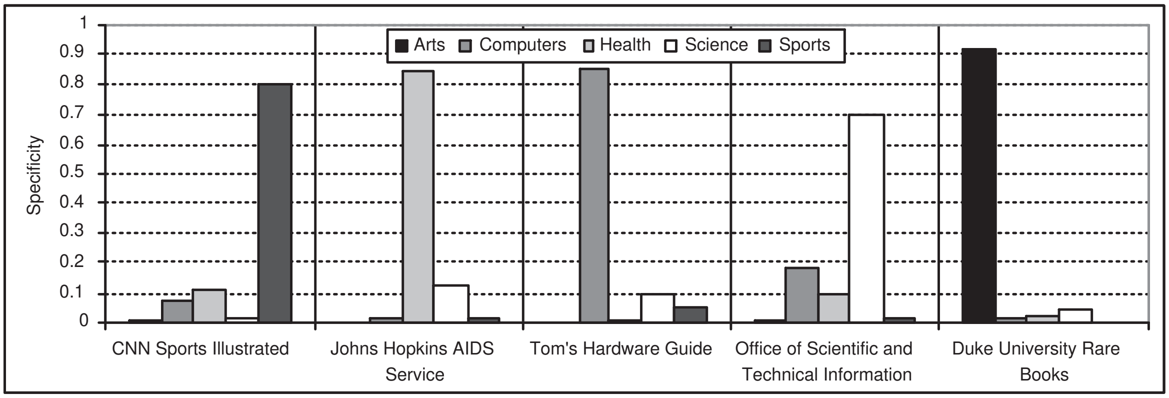

Figure 2: Distribution of documents in the top-level categories for five searchable web databases.

in [8]. (See [2, 3, 10, 6, 7] for other related work relevant to database classification.) As we discussed in the introduction, our technique can be also applied to the classification of any database that offers a search interface for its contents, no matter if its contents are "hidden" or not.

3.2 Crawling-based Classification

As described in Section 2, the categorization of a database is determined by its distribution of documents across categories. We now describe a classification approach that categorizes a crawlable database by simply retrieving its documents using a web crawler and classifying them with a previously-trained document classifier. This algorithm works as follows to classify a web database:

1. Train a rule-based document classifier with a set of preclassified documents.

2. Using a crawler, download all documents from the web database.

3. Classify each retrieved document into a set of categories using the document classifier from Step 1.

4. Classify the database using the number of documents classified into each category from Step 3.

Note that this crawling-based classification approach could be applied both to web sites that are crawlable through "traditional" crawlers (e.g., [4]) and to hidden-web databases as novel crawlers for them are developed [9].

4 Experimental Evaluation

To compare the accuracy and efficiency of the query- and crawling-based classification approaches, we use the five real web databases with crawlable content that are listed in Table 1. For evaluation purposes, we manually classified the five databases by inspecting their contents.

To implement the query-based algorithm of Section 3.1, we constructed a simple "wrapper" around each database. Given a keyword-based query, the wrapper sends it to the database, and extracts *only* the number of matches that the query produces. For each database, we measured the classification accuracy in terms of precision and recall, the amount of information that was transmitted over the network, and the time needed to complete the classification process.

URL	*Brief Description*	*Category*
`http://www.cnnsi.com/`	CNN Sports Illustrated	Sports
`http://www.tomshardware.com/`	Tom's Hardware Guide	Computers
`http://hopkins-aids.edu/`	Johns Hopkins AIDS Service	AIDS
`http://odyssey.lib.duke.edu/`	Duke University Rare Books	Literature
`http://www.osti.gov/`	Office of Scientific and Technical Information	Science

Table 1: The searchable web databases used in the experiments.

	Crawling-based Classification			*Query-based Classification*		
Database	*Time*	*Files*	*Size*	*Time*	*Queries*	*Size*
CNN Sports Illustrated	1325 min	270,202	8 Gb	2 min (-99.8%)	112	357 Kb (-99.9%)
Tom's Hardware Guide	32 min	2,928	105 Mb	3 min (-90.6%)	292	602 Kb (-99.7%)
Johns Hopkins AIDS Service	13 min	1,823	17 Mb	1 min (-92.3%)	314	723 Kb (-95.7%)
Duke University Rare Books	2 min	3,242	16.5 Mb	3 min (+50.0%)	397	1012 Kb (-93.8%)
Office of Scientific and Technical Information	210 min	30,749	416 Mb	2 min (-99.0%)	174	423 Kb (-99.8%)

Table 2: The performance of crawling- and query-based classification for five databases.

To implement the crawling-based algorithm of Section 3.2, we used the GNU Foundation's *wget* tool [8] to crawl and download the contents of each database. We classify each downloaded page *individually* using the same document classifier that the query-based approach needs to generate query probes. Hence at each stage of the crawling process, we know the category distribution of the pages already downloaded, from which we can derive a preliminary classification of the database. We measured the classification accuracy in terms of precision and recall at different crawling stages, to examine the speed with which the crawling-based approach reached the correct classification decision. Additionally, we measured the amount of information that was transmitted over the network, and the time needed to complete the classification process.

Table 2 lists the evaluation results for the two approaches over the databases in our data set. Except for one case, the time needed to complete the classification process and the amount of information transmitted over the network was orders of magnitude larger for the crawling-based approach. Additionally, the crawling-based approach needed extra local processing power to locally classify the documents. When the number of retrieved pages is large, this can create a significant local overhead.

One significant advantage of the query-based approach is the fact that its execution time is largely independent of the size of the database, so this approach can scale to databases of virtually any size. Additionally, no documents are retrieved during querying, which speeds up the classification algorithm and minimizes the required bandwidth for the classification. Finally, the query-based approach gives a better bound on completion time, since the maximum number of queries sent to a database depends only on the given classification scheme and is usually small. In contrast, a crawler might spend a lot of time crawling a large site in order to determine a final classification.

One question that remains to be answered is whether the crawling-based approach can be improved by requiring only a small portion of a site to be downloaded. To understand how fast the crawling approach can reach the correct classification, we measured the precision and recall of the classification results when different fractions of the database are crawled and classified. Our experiments reveal that often a significant fraction of a web database may need to be retrieved before a correct classification decision can be made. For example, as can be inferred from Table 3, the crawler started crawling parts of the CNN.SI that were about specific sports (cycling in this case). Consequently, early in the crawling process, the category distribution was wrongly biased towards this sport and the site was incorrectly classified under this category, rather than being classified under the more general "Sports" category. Only after crawling 70% of all pages did this approach yield the right category

[8] `http://www.gnu.org/software/wget/wget.html`

% Crawled	**10%**	**30%**	**50%**	**60%**	**70%**	**100%**
CNN Sports Illustrated	Cycling Multimedia $P = 0.5$ $R = 0.09$	Cycling Multimedia $P = 0.5$ $R = 0.09$	Cycling Multimedia $P = 0.5$ $R = 0.09$	Cycling $P = 1.0$ $R = 0.09$	**Sports** $P = 1.0$ $R = 1.0$	**Sports** $P = 1.0$ $R = 1.0$
Tom's Hardware Guide	**Computers** **Rock** $P = 0.91$ $R = 1.0$	**Computers** **Rock** $P = 0.91$ $R = 1.0$	**Computers** **Rock** $P = 0.91$ $R = 1.0$	**Computers** **Rock** $P = 0.91$ $R = 1.0$	**Computers** **Rock** $P = 0.91$ $R = 1.0$	**Computers** **Rock** $P = 0.91$ $R = 1.0$
Johns Hopkins AIDS Service	**AIDS** $P = 1.0$ $R = 1.0$	**AIDS** $P = 1.0$ $R = 1.0$	**AIDS** $P = 1.0$ $R = 1.0$	**AIDS** $P = 1.0$ $R = 1.0$	**AIDS** $P = 1.0$ $R = 1.0$	**AIDS** $P = 1.0$ $R = 1.0$
Duke University Rare Books	Poetry Texts Classics History Photography $P = 0.6$ $R = 0.6$	Poetry $P = 1.0$ $R = 0.2$	**Poetry** **Texts** $P = 1.0$ $R = 0.4$	**Poetry** **Texts** $P = 1.0$ $R = 0.4$	**Poetry** **Texts** $P = 1.0$ $R = 0.4$	**Poetry** **Texts** $P = 1.0$ $R = 0.4$
Office of Scientific and Technical Information	Biology $P = 1.0$ $R = 0.33$	Root $P = 0.25$ $R = 1.0$	Root $P = 0.25$ $R = 1.0$	**Biology** $P = 1.0$ $R = 0.33$	**Biology** $P = 1.0$ $R = 0.33$	**Biology** $P = 1.0$ $R = 0.33$

Table 3: The crawling-based classification and associated precision (P) and recall (R) for five databases after crawling different fractions of each database (specificity threshold $\tau_s = 0.4$).

for this database. A similar observation holds for *Duke's Rare Book Collection* and the *Office of Scientific and Technical Information*. On the other hand, the classification of *Tom's Hardware Guide* and the *Johns Hopkins AIDS Service* was remarkably accurate in the crawling process and converged to the correct result very fast. This happened because these two sites contain documents that are relatively homogeneous in topic. Hence, even the first few pages retrieved were good representatives of the database as a whole.

In summary, the crawling-based approach is prone to producing wrong classification decisions at early stages of the crawling. A crawling-based approach could produce reasonable classification results early on only if crawling could somehow guarantee that the documents crawled first reflected the real topic distribution in the entire database. However, currently no crawler can perform such a traversal and it is doubtful that any will ever be able to do so, since to build such a crawler presupposes knowledge of the distribution of documents at the sites. In contrast, our query-based method manages to detect the correct classification of all these databases using only a fraction of the time and data required for the crawling-based approach. Hence, these first experimental results suggest that the query-based approach is a better alternative for the classification of web databases, both in terms of classification accuracy and efficiency.

5 Conclusion

In this paper we have summarized a query-based classification for searchable web databases [8]. This algorithm is the state-of-the-art for the classification of databases with "uncrawlable" content. Interestingly, the algorithm can also be used for crawlable databases, as long as they expose a search interface. In such cases, we could alternatively classify the databases by crawling their contents and classifying the retrieved documents. We have argued in this paper that our query-based approach significantly outperforms the crawling-based approach. The query-based approach is in some cases orders of magnitude more efficient than its crawling-based counterpart both in execution time and in utilization of network resources. Additionally, the query-based algorithm reaches the correct classification decision quickly, while the crawling-based approach depends greatly on how well the crawling order reflects the actual topic distribution in the database. Please refer to `http://qprober.cs.columbia.edu` for more details about our query-based algorithm, as well as to access a prototype system demonstrating our approach.

Acknowledgments

Luis Gravano and Panagiotis G. Ipeirotis were funded in part by the National Science Foundation (NSF) under Grants No. IIS-97-33880 and IIS-98-17434.

References

[1] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, May 2001.

[2] James P. Callan, Margaret Connell, and Aiqun Du. Automatic discovery of language models for text databases. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data (SIGMOD'99)*, pages 479–490, 1999.

[3] Jamie Callan and Margaret Connell. Query-based sampling of text databases. *ACM Transactions on Information Systems*, 19(2):97–130, 2001.

[4] Junghoo Cho, Héctor García-Molina, and Lawrence Page. Efficient crawling through URL ordering. In *Proceedings of the Seventh International World Wide Web Conference (WWW7)*, 1998.

[5] William W. Cohen. Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning (ICML'95)*, pages 115–123, 1995.

[6] Ron Dolin, Divyakant Agrawal, and Amr El Abbadi. Scalable collection summarization and selection. In *Proceedings of the Fourth ACM International Conference on Digital Libraries (DL'99)*, pages 49–58, 1999.

[7] Susan Gauch, Guijun Wang, and Mario Gomez. ProFusion*: Intelligent fusion from multiple, distributed search engines. *The Journal of Universal Computer Science*, 2(9):637–649, September 1996.

[8] Panagiotis G. Ipeirotis, Luis Gravano, and Mehran Sahami. Probe, count, and classify: Categorizing hidden-web databases. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data (SIGMOD 2001)*, pages 67–78, 2001.

[9] Sriram Raghavan and Héctor García-Molina. Crawling the hidden web. In *Proceedings of the 27th International Conference on Very Large Databases (VLDB 2001)*, 2001.

[10] Wenxian Wang, Weiyi Meng, and Clement Yu. Concept hierarchy based text database categorization in a metasearch engine environment. In *Proceedings of the First International Conference on Web Information Systems Engineering (WISE'2000)*, pages 283–290, 2000.