Automatically Constructing a Directory of Molecular Biology Databases

Luciano Barbosa, Sumit Tandon, and Juliana Freire

School of Computing, University of Utah

Abstract. There has been an explosion in the volume of biology-related information that is available in online databases. But finding the *right* information can be challenging. Not only is this information spread over multiple sources, but often, it is hidden behind form interfaces of online databases. There are several ongoing efforts that aim to simplify the process of finding, integrating and exploring these data. However, existing approaches are not scalable, and require substantial manual input. Notable examples include the NCBI databases and the NAR database compilation. As an important step towards a scalable solution to this problem, we describe a new infrastructure that automates, to a large extent, the process of locating and organizing online databases. We show how this infrastructure can be used to automate the construction and maintenance of a Molecular Biology database collection. We also provide an evaluation which shows that the infrastructure is scalable and effective—it is able to efficiently locate and accurately identify the relevant online databases.

1 Introduction

Due to the explosion in the number of online databases, there has been increased interest in leveraging the high-quality information present in these databases [1, 2, 7, 10, 19]. However, finding the right databases can be challenging. For example, if a biologist needs to locate databases related to molecular biology and searches on Google for the keywords "molecular biology database" over 27 million documents are returned. Among these, she will find pages that contain databases, but the results also include a very large number of pages from journals, scientific articles, etc.

Recognizing the need for better mechanisms to locate online databases, there have been a number of efforts to create online database collections such as the NAR database compilation [9], a manually created collection which lists databases of value to biologists. Given the dynamic nature of the Web, where new sources are constantly added, manual approaches to create and maintain database collections are not practical. But automating this process is non-trivial. Since online databases are sparsely distributed on the Web, an efficient strategy is needed to locate the forms that serve as entry points to these databases. In addition, online databases do not publish their schemas and since their contents

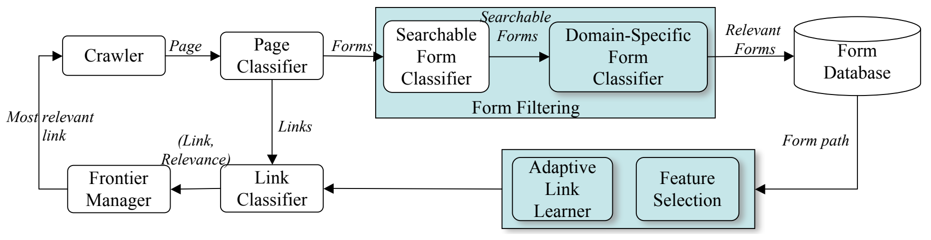


Fig. 1. Architecture of *ACHE*.

are often hidden behind form interfaces, they are hard to retrieve. Thus, a scalable solution must determine the relevance of a form to a given database domain just by examining information that is available in and around forms.

In previous work [2–4], we proposed *ACHE* (**A**daptive **C**rawler for **H**idden-Web **E**ntry Points), a new scalable framework that addresses these problems. We showed, experimentally, that *ACHE* is effective for a representative set of commercial databases. In this paper, we describe a case study we carried out to investigate the effectiveness of this framework for different domains, and in particular, for non-commercial online databases. We chose to focus on databases related to molecular biology for two key reasons: these are often academic databases; and there is already a sizeable collection of these databases [9] which can serve as a basis for comparison.

The remainder of the paper is organized as follows. In Section 2, we give a brief overview of the *ACHE* framework. In Section 3, we describe in detail the process we followed to customize *ACHE* to the molecular biology domain. We discuss the issues we faced in the process, and show that, because the different components of *ACHE* use learning-based techniques, they can be easily adapted to a different domain. We present our experimental evaluation in Section 4. The results indicate that *ACHE* is effective: it is able to efficiently locate and accurately identify online databases related to molecular biology. We conclude in Section 6, where we outline directions for future work.

2 Searching and Identifying Online Databases

ACHE provides an end-to-end solution to the problem of locating and organizing online databases. The high-level architecture of the system is shown in Figure 1. *ACHE* uses a focused crawler to locate online databases. Similar to topic-specific crawlers, *ACHE* also uses Web page contents to focus its search on a given topic. But to deal with the sparseness of online databases on the Web, it prioritizes links that are more likely to lead to forms in the database domain sought. *ACHE* also uses a form-filtering process to select the relevant forms among the set of forms retrieved by the crawler. This form-filtering process is required because even a focused crawler invariably retrieves a diverse set of forms, including searchable forms (i.e., forms used to search over a database) from multiple database domains, and non-searchable forms that do not represent database queries such as, for example, forms for login, mailing list subscriptions, Web-based email forms.

Consider for example, the Form-Focused Crawler (FFC) [2] which is optimized for locating searchable Web forms. For a set of representative database domains, on average, only 16% of the forms retrieved by the FFC are actually relevant—for some domains this percentage can be as low as 6.5%. These numbers are even lower for less focused crawlers [6, 8].

In what follows, to make this paper self-contained, we briefly describe the components of *ACHE*. For a detailed description, the reader is referred to [3, 4].

2.1 Searching for Online Databases

Each page retrieved by the crawler is sent to the *Page Classifier*, which is trained to identify pages that belong to a particular topic based on their contents. It uses the same strategy as the best-first crawler of [6]. The page classifier analyzes a page P and assigns to it a score which reflects the probability that P belongs to the focus topic. A page is considered relevant if this probability is greater than a certain threshold (0.5 in our case).

If a page is determined to be relevant, its links are extracted and used as inputs to the *Link Classifier*.

The Link Classifier learns to estimate the distance between a link and a target page based on link patterns: given a link, the link classifier assigns a score to the link which corresponds to the estimated distance between the link and a page that contains a relevant form. The Frontier Manager uses this estimate to prioritize *promising* links, including links that have *delayed benefit*—links which belong to paths that will *eventually* lead to pages that contain searchable forms. As we discuss in [3], considering links with delayed benefit is essential to obtain high harvest rates while searching for sparse concepts such as online databases on the Web. Since searchable forms are sparsely distributed on the Web, by prioritizing only the links that bring immediate return, i.e., links whose patterns are similar to those of links pointing to pages containing searchable forms, the crawler may miss target pages that can only be reached with additional steps.

The Link Classifier is constructed as follows. Given a set of URLs of pages that contain forms in a given database domain, paths to these pages are obtained by crawling backwards from these pages. *ACHE* uses two different approximations of the Web graph to perform a backward crawl: it uses the `link:` facility provided by search engines [5] at the beginning of the crawling process; and it uses the Web subgraph collected during the crawler execution. The backward crawl proceeds in a breadth-first manner. Each level *l+1* is constructed by retrieving all documents that point to the documents in level *l*. From the set of paths gathered, the *best* features of the links are automatically selected. These features consist of the highest-frequency terms extracted from text in the neighborhood of the link, as well as from the URL and anchor. Using these features, the classifier is trained to estimate the distance between a given link (from its associated features) and a target page that contains a searchable form. Intuitively, a link that matches the features of level *1* is likely to point to a page that contains a form; and a link that matches the features of level l is likely l steps away from a page that contains a form.

The *Frontier Manager* controls the order in which pages are visited. It creates one queue for each level of the backward crawl. Links are placed on these queues based on their similarity to the features selected for the corresponding level of the link classifier. Intuitively, the lower the level of the link classifier, the higher is the priority of the queue. When the crawler starts, all seeds are placed in queue 1. At each crawling step, the crawler selects the link with the highest relevance score from the first non-empty queue. If the page it downloads belongs to the target topic, its links are classified by link classifier and added to the most appropriate queue.

The focused crawler learns new link patterns during the crawl and automatically adapts its focus based on these new patterns. As the crawler navigates through Web pages, successful paths are gathered, i.e., paths followed by the crawler that lead to relevant forms. Then, the *Feature Selection* component automatically extracts the patterns of these paths. Using these features and the set of path instances, the *Adaptive Link Learner* generates a new Link Classifier that reflects these newly-learned patterns.[1] The Adaptive Link Learner is invoked periodically, after the crawler visits a pre-determined number of pages. Experiments over real Web pages in a representative set of commercial domains showed that online learning leads to significant gains in harvest rates—the adaptive crawler retrieve up to three times as many forms as a crawler that use a fixed focus strategy [3].

2.2 Identifying Relevant Databases

The *Form Filtering* component is responsible for identifying relevant forms gathered by *ACHE*, and it does so by examining the visible content in the forms. The overall performance of the crawler is highly-dependent on the accuracy of the form filtering process, which assists *ACHE* in obtaining high-quality results and also enables the crawler to adaptively update its focus strategy. If the Form Filtering process is inaccurate, crawler efficiency can be greatly reduced as it drifts way from its objective through unproductive paths.

Instead of using a single, complex classifier, our form filtering process uses a sequence of simpler classifiers that learn patterns of different subsets of the form feature space [4]. The first is the *Generic Form Classifier* (*GFC*), which uses structural patterns to determine whether a form is searchable. Empirically, we have observed that these structural characteristics of a form are a good indicator as to whether the form is searchable [2]. The second classifier in the sequence identifies searchable forms that belong to a given domain. For this purpose, we use a more specialized classifier, the *Domain-Specific Form Classifier* (*DSFC*). The DSFC uses the textual content of a form to determine its domain. Intuitively, the form content is often a good indicator of the database domain—it contains metadata and data that pertain to the database.

[1] The length of the paths considered depends on the number of levels used in the link classifier.

By partitioning the feature space of forms, not only can simpler classifiers be constructed that are more accurate and robust, but this also enables the use of learning techniques that are more effective for each feature subset. Whereas decision trees [14] gave the lowest error rates for determining whether a form is searchable based on structural patterns, SVM [14] proved to be the most effective technique to identify forms that belong to the given database domain based on their textual content.

3 Constructing the Molecular Biology Database Directory

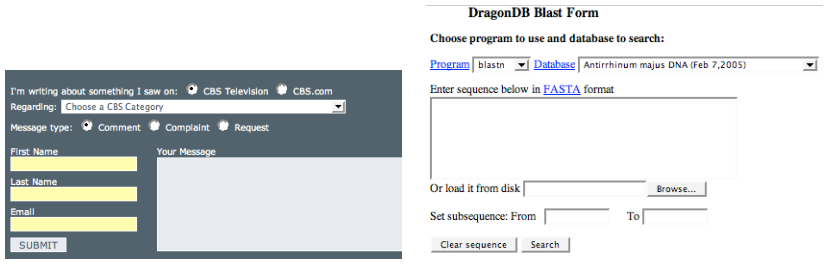
In this section we describe the process we followed to build a collection of molecular biology online databases. This process consists of customizing three components of the *ACHE* framework: Page Classifier, Link Classifier and Form Filtering.

Page Classifier. The Page Classifier defines the broad search topic for the crawler: based on the page content (words in the page), the Page Classifier predicts whether a given page belongs to a topic or not. We used Rainbow [12], a freely-available Naïve Bayes classifier, to build the Page Classifier. To train it, we crawled the biology-related Web sites listed in `dmoz.org` and gathered 2800 pages to serve as positive examples. Because of the great variety of pages that can be visited during the crawl, constructing a set of representative negative examples is more challenging. To select negative examples, we ran the crawler with the selected positive examples and an initial set of negative examples taken from a corpus that comes with the Rainbow classifier. We then added the misclassified pages to the set of negative examples. Examples of such misclassified pages included non-English pages and pages from foreign porn sites. A total of 4671 negative examples were collected.

The Page Classifier was then constructed using the 50 terms that led to the highest information gain. For the Molecular Biology domain, these terms included: biology, molecular, protein, genome, ncbi, length, substring, structure, gene, genomics, nih, parent, sequence, pubmed, entrez, nlm, fellows, postdoctoral, research, dna.

Link Classifier. We created the Link Classifier from a backward crawl of depth 3. The set of seeds chosen to train the Link Classifier comprised 64 relevant Web forms manually selected from NAR collection. For each of the feature spaces of links (url, anchor and text in link neighborhood), the 5 most frequent words are selected. To build the classifier we used WEKA [18], an open source data mining tool. The classification algorithm (Naïve Bayes) is used to estimate the probabilities of a link being 1, 2, or 3 steps away from a form page.

Form Filtering. As discussed above, the Form Filtering uses two classifiers: the GFC (based on form structure) and DSFC (based on form content). In our initial experiment, we used the GFC we had constructed for identifying searchable commercial databases [4]. An inspection of the misclassified forms showed that


I'm writing about something I saw on: CBS Television CBS.com
Regarding: Choose a CBS Category
Message type: Comment Complaint Request
First Name
Last Name
Email
SUBMIT
Your Message
DragonDB Blast Form
Choose program to use and database to search:
Program blastn Database Antirrhinum majus DNA (Feb 7,2005)
Enter sequence below in FASTA format
Or load it from disk Browse...
Set subsequence: From To
Clear sequence Search

(a) Non-searchable form (b) Blast form

Fig. 2. Similarity of a non-searchable form in a commercial domain and a searchable form in the molecular biology domain.

some searchable forms in the molecular biology domain are structurally similar to non-searchable forms of commercial sites. The presence of features such as text areas, buttons labeled with the string "submit", file inputs, are good indicators that a (commercial) form is non-searchable. However, these features are also present in many searchable forms in the molecular biology domain (e.g., Blast search forms). This is illustrated in Figure 2. To address this problem, we added a sample of the misclassified forms to the pool of positive examples, and generated a new instance of the classifier. The GFC was then able to correctly classify forms like the Blast forms as searchable—its accuracy improved to 96%.

To generate the DSFC, we manually gathered 150 positive examples of forms from the NAR collection [9]. The negative examples were obtained as follows: we ran the crawler and filtered the searchable forms using the GFC; then, from these searchable forms we manually selected 180 forms that did not belong to the molecular biology domain. These forms included, e.g., forms related to chemistry and agriculture, as well forms for searching for authors and journals related to molecular biology. Using these training examples, we generated the first version of DSFC. This version, however had a very low precision: only 16%. The problem was due to false positives. Unlike the commercial domains, the crawler retrieved a large number of non-English pages. As the DSFC was not trained to handle non-English terms it incorrectly classified these forms as relevant. After we added the misclassified (non-English) forms to the set of negative examples, the accuracy of the DSFC increased substantially, to 65%. To try and further improve the accuracy, we added additional false positives misclassified by the second version of the DSFC to the pool of negative examples and, once again, constructed a new instance of the classifier. The third version obtained 89% accuracy. The top 20 terms used to build the DSFC were: name, type, select, result, keyword, gene, sequenc, databa, enter, ani, option, page, help, titl, protein, number, advanc, onli, format, word.

4 Experimental Evaluation

In this section we first assess the effectiveness of *ACHE* for constructing a high-quality set of molecular biology databases. We then compare our results with those of the manually constructed NAR collection.

	Recall	**Specificity**
Adaptive	0.82	0.96

Table 1. Quality measurement for GFC.

	Recall	**Precision**	**Accuracy**
Adaptive	0.73	0.93	0.96

Table 2. Quality measurement for Form Filtering (GFC+DSFC).

To verify the effectiveness of the adaptive crawler in this domain, we executed the following crawler configurations:

- Baseline Crawler: A variation of the best-first crawler [6]. The page classifier guides the search and the crawler follows all links of a page whose contents are classified as being on-topic;
- Static Crawler: Crawler operates using a fixed policy which remains unchanged during the crawling process;
- Adaptive Crawler: ACHE starts with a pre-defined policy, and this policy is dynamically updated after crawling 10,000 pages.

All configurations were run using 35 seeds obtained from `dmoz.org` and crawled 100,000 pages. The Link Classifier was configured with three levels.

Since the goal of *ACHE* is to find *relevant* online databases in the molecular biology domain, we measured the effectiveness of the crawler configurations in terms of the total number of relevant forms gathered. We manually inspected the output of Form Filtering to calculate the values for: accuracy, recall; precision and specificity. *Accuracy* is a suitable measure when the input to the classifier contains similar proportions of positive and negatives examples; *recall* captures the number of relevant items retrieved as fraction of all relevant items; *precision* represents the number of relevant items as a fraction all the items predicted as positive by the classifier; and *specificity* is the proportion of actual irrelevant items predicted as irrelevant.

The results obtained by the GFC (see Table 1) confirm that it can identify most of the relevant forms (high recall) and to filter out most of the irrelevant forms (high specificity). As Table 2 shows, the combination of the GFC and DSFC leads to a very high recall, precision and accuracy. This indicates that the Form Filtering process is effective and that a high-quality (homogeneous) collection of databases can be generated by *ACHE*.

Figure 3 shows the number of relevant forms retrieved by the three crawler configurations over time. The Adaptive Crawler outperforms both the Static and the Baseline configurations, retrieving 513 relevant forms after crawling 100,000 pages versus 341 and 376 relevant forms retrieved by Static and Baseline, respectively. This shows that the feedback from Form Filtering is effective in boosting the crawler performance. Table 3 shows the features used by the initial Link Classifier and the features learned during the crawl that are used by the final classifier. A similar behavior has been observed for crawls over commercial domains [3]. This indicates that the adaptive strategy is effective regardless of the domain.

Field	**Initial features**	**Final features**
URL	link, search, full, genom, index	search, blast, genom, form, bioinfo
Anchor	data, genom, for, text, full	search, blast, gene, databas, sequenc
Around	bio, data, info, genom, gene	search, databas, gene, genom, sequenc

Table 3. Features of the Link Classifiers used at the beginning and at the end of the crawl process.

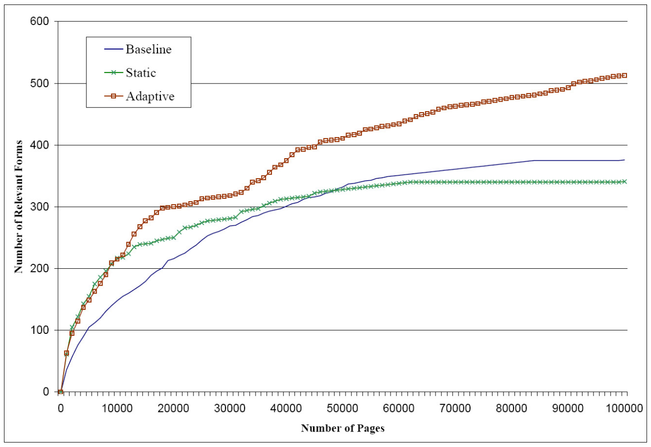


Fig. 3. Behavior of different crawler configurations over time.

NAR Collection. The NAR collection lists 968 databases. But their concept of databases is more generic than ours: they consider as a database both pages that contains tables with information about genes and proteins, and pages that contain forms (or links to pages that contain forms). In our evaluation, we consider only the searchable forms accessible through the NAR collection. To extract the searchable forms directly or indirectly accessible through the NAR collection, we crawled the links provided (using `wget` with depth 1). Among the 20,000 pages retrieved by `wget`, 700 relevant forms were identified. Although *ACHE* obtained 513 forms, we should note that the NAR collection has been maintained for over 7 years—the earliest reference we found dates back to 1999—and it has become a very popular resource. Once *ACHE* was configured, the 513 forms were automatically gathered in 4 hours. This shows that such a collection can be efficiently maintained over time. In addition, these forms were obtained in a relatively small crawl (only 100,000 pages). The positive slope for the Adaptive Crawler graph in Figure 3 indicates that additional forms can be obtained in larger crawls. This is an issue we plan to investigate in future work.

5 Related Work

BioSpider [11] is a system that integrates biological and chemical online databases. Given a biological or chemical identifier, BioSpider produces a report containing physico-chemical, biochemical and genetic information about the identifier. Although the authors mention BioSpider performs a crawl to locate the underlying sources, no details are given about the crawling process. Also, the number of sources they integrate is very small—only about 20 databases are listed on their Web site.

Ngu et al. [15] proposed an approach to classify search interfaces by probing these interfaces and trying to match the control flow of the interface against a standard control flow. Thus, for a specific type of form (which they refer to as a service class), e.g., a Blast search, they create a corresponding flow graph pattern from a sample of known interfaces and try to match new interfaces against that pattern. An important limitation of this solution comes from its reliance on the ability to automatically fill out structured forms. The difficulties in automatically filling out structured Web forms are well-documented in the literature [7, 16].

InfoSpiders [17] is a multi-agent focused crawler specialized for biomedical information whose goal is to fetch information about diseases when given information about genes. A study by Menczer et al. [13] comparing several topic-driven crawlers (including InfoSpiders) found that the best-first approach (the Baseline configuration in Section 4) leads to the highest harvest rate among the crawlers in the study. As we discuss in Section 4, our adaptive crawler outperforms the best first crawler by a large margin.

6 Conclusion and Discussion

In this paper we described a case study we carried out to evaluate the extensibility and effectiveness of the *ACHE* framework for constructing a high-quality online database directories. We described the process of customizing the framework for molecular biology databases; and performed an evaluation which showed that *ACHE* is able to efficiently locate and accurately identify databases in this domain. The number of relevant forms automatically gathered (after a 4-hour crawl) is very close to the number of forms listed in a manually created collection that has been maintained for over 7 years. This indicates that *ACHE* provides a scalable solution to the problem of automatically constructing high-quality, topic-specific online database collections. These results also reinforce our choice of applying learning techniques. Because we use learning classifiers in the different components of *ACHE*, with some modest tuning, the system can be customized for different domains.

It is well-known, however, that the performance of machine learning techniques, such as the classifiers used in our framework, is highly-dependent on the choice of training examples used to construct them. And building a representative sample of forms is difficult due to the large variability in form content and structure, even within a well-defined domain. We are currently investigating strategies that simplify the process of gathering positive and negative examples.

To help users locate relevant databases, we are designing intuitive and expressive query interfaces that support both simple keyword-based queries and structured queries (e.g., find forms that contain an attribute with a given label). Although our focus has been on databases accessible through forms, in future work we plan to investigate extensions to our infrastructure for handling more general notions of online databases, such as for example, pages that contain tables with biology-related information.

Acknowledgments. This work was partially supported by the National Science Foundation and a University of Utah Seed Grant.

References

1. L. Barbosa and J. Freire. Siphoning Hidden-Web Data through Keyword-Based Interfaces. In *Proceedings of SBBD*, pages 309–321, 2004.
2. L. Barbosa and J. Freire. Searching for Hidden-Web Databases. In *Proceedings of WebDB*, pages 1–6, 2005.
3. L. Barbosa and J. Freire. An Adaptive Crawler for Locating Hidden-Web Entry Points. In *Proceedings of WWW*, 2007. To appear.
4. L. Barbosa and J. Freire. Combining Classifiers to Organize Online Databases. In *Proceedings of WWW*, 2007. To appear.
5. K. Bharat, A. Broder, M. Henzinger, P. Kumar, and S. Venkatasubramanian. The connectivity server: Fast access to linkage information on the Web. *Computer Networks*, 30(1-7):469–477, 1998.
6. S. Chakrabarti, M. van den Berg, and B. Dom. Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery. *Computer Networks*, 31(11-16):1623–1640, 1999.
7. K. C.-C. Chang, B. He, and Z. Zhang. Toward Large-Scale Integration: Building a MetaQuerier over Databases on the Web. In *Proceedings of CIDR*, pages 44–55, 2005.
8. M. Diligenti, F. Coetzee, S. Lawrence, C. L. Giles, and M. Gori. Focused Crawling Using Context Graphs. In *Proceedings of VLDB*, pages 527–534, 2000.
9. M. Galperin. The molecular biology database collection: 2007 update. *Nucleic Acids Res*, 35, 2007.
10. W. Hsieh, J. Madhavan, and R. Pike. Data management projects at Google. In *Proceedings of ACM SIGMOD*, pages 725–726, 2006.
11. C. Knox, S. Shrivastava, P. Stothard, R. Eisner, and D. S. Wishart. BioSpider: A Web Server for Automating Metabolome Annotations. In *Pacific Symposium on Biocomputing*, pages 145–156, 2007.
12. A. McCallum. Rainbow. http://www-2.cs.cmu.edu/ mccallum/bow/rainbow/.
13. F. Menczer, G. Pant, M. Ruiz, and P. Srinivasan. Evaluating topic-driven Web crawlers. In *Proceedings of SIGIR*, pages 241–249, 2001.
14. T. Mitchell. *Machine Learning*. McGraw Hill, 1997.
15. A. H. Ngu, D. Rocco, T. Critchlow, and D. Buttler. Automatic discovery and inferencing of complex bioinformatics web interfaces. *World Wide Web*, 8(3):463–493, 2005.
16. S. Raghavan and H. Garcia-Molina. Crawling the Hidden Web. In *Proceedings of VLDB*, pages 129–138, 2001.
17. P. Srinivasan, J. Mitchell, O. Bodenreider, G. Pant, and F. Menczer. Web Crawling agents for Retrieving Biomedical Information. In *Workshop on Agents in Bioinformatics (NETTAB-02)*, 2002.
18. I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.
19. W. Wu, C. Yu, A. Doan, and W. Meng. An Interactive Clustering-based Approach to Integrating Source Query interfaces on the Deep Web. In *Proceedings of ACM SIGMOD*, pages 95–106, 2004.