Efficient Web Harvesting Strategies for Monitoring Deep Web Content

Mohammad Khelghati
Database Group, University of Twente, Netherlands
s.m.khelghati@utwente.nl

Djoerd Hiemstra
Database Group, University of Twente, Netherlands
d.hiemstra@utwente.nl

Maurice van Keulen
Database Group, University of Twente, Netherlands
m.vankeulen@utwente.nl

Abstract

Web content changes rapidly [18]. In *Focused Web Harvesting* [17] which aim it is to achieve a complete harvest for a given topic, this dynamic nature of the web creates problems for users who need to access a set of all the relevant web data to their topics of interest. Whether you are a fan following your favorite idol or a journalist investigating a topic, you may need not only to access all the relevant information but also the recent changes and updates. General search engines like Google apply several techniques to enhance the freshness of their crawled data. However, in *focused web harvesting*, we lack an efficient approach that detects changes for a given topic over time. In this paper, we focus on techniques that can keep the relevant content to a given query up-to-date. To do so, we test four different approaches to efficiently harvest all the changed documents matching a given entity by querying web search engines. We define a document with changed content or a newly created or removed document as a changed document. Among the proposed change detection approaches, the *FedWeb* method outperforms the other approaches in finding the changed content on the web for a given query with 20 percent, on average, better performance.

1. INTRODUCTION

Web content is constantly and rapidly changing [18, 11, 7, 10, 22]. This dynamic nature of web data creates difficulties for users who need to access a complete collection of documents relevant to their topics of interest. For example, as a fan following your favorite idol or a business analyst studying a stock market, you want to access not only *all* but also the *latest* information relevant to your topics of interest. You need to keep your collection up-to-date over time.

To monitor data, first, we need a complete coverage for a given topic. To reach this complete coverage, accessing a crawl of the whole web is an optimum solution. However, accessing such a crawl seems impracticable even for big companies and governments. As an alternative, *focused web harvesting* techniques are applicable. Khelghati *et al.* define focused web harvesting as harvesting all documents matching a given entity by querying a web search engine in [17]. For instance, information about “Bernie SandersâĂİ, “Islamic StateâĂİ or “Golden Ball AwardâĂİ are retrieved from indexed data in general search engines or hidden data behind web forms by submitting a stream of queries and retrieving their returned results. Queries are formed by adding terms to a seed query with the goal of returning more unique documents with respect to the imposed limitations by search engines on the number of submitted queries by a user and the number of returned results he can view [17].

These imposed limitations by search engines also affect having an up-to-date data collection. Through applying focused web harvesting techniques, users can collect relevant information to their topics of interest and monitor its changes over time by re-issuing queries. However, considering the imposed limitations by search engines, we need efficient approaches to detect the changed content on the web without recollecting all the previously harvested documents. We need approaches that facilitate efficient re-harvesting processes. To attain these approaches, we study the applied techniques in crawlers and big search engines to increase the freshness of their indices. We also explore their algorithms for detection of changed, duplicate and near-duplicate pages. We investigate deep web harvesting techniques, especially focused web harvesting, to find methods applicable in detecting changed content on the web for a given topic.

Based on these studies, we propose approaches that are based on adapting a query generation mechanism to return more desirable documents. This work proposes solutions for efficient retrieval of changed content on the web in the domain of focused web harvesting.

Contributions.

As our main contribution, we study four different methods to find the most efficient approach to retrieve changed documents on the web that are relevant to a given topic. We test these approaches on our test search engine and report the results. The results show that we can improve the retrieved changed content by at least 20 percent by submitting the same number of queries but the ones that are more likely to occur in changed documents.

Outlook.

In Section 2, we study the available solutions for accessing and monitoring deep web data. In Section 3, the proposed

iiWAS '16, November 28-30, 2016, Singapore, Singapore
© 2016 ACM. ISBN 978-1-4503-4807-2/16/11…$15.00
DOI: http://dx.doi.org/10.1145/3011141.3011198

techniques for efficient re-harvesting are introduced. The results of testing these methods on our test set is presented in Section 4. Section 5 draws conclusions and discusses further future work.

2. LITERATURE STUDY

Recent studies on accessing deep web data mainly focus on harvesting websites requiring form submissions [2, 20, 13, 8], studying the web forms to achieve an efficient data access approach. For example, Madhavan *et al.* [20] investigate how to identify the values and types of different web form fields and how to efficiently navigate the search space of possible form input combinations to avoid unnecessary submissions leading to a more efficient source sampling process. Although the goal of these studies is not entity focused harvesting, the idea of devising different query generation plans in querying a data source is relevant. These studies focus on the features of forms and, require additional form analysis and extra domain knowledge.

A deep web data access is mainly performed through *query based sampling* referred to as *QBS*. In QBS, by sending a query to a search engine, the set of returned results is considered as a sample of documents [5]. To form these queries, *query expansion* techniques are applied. Query expansion is the process of reformulating a given query with the goal of getting a better query, a query that is more likely to retrieve relevant documents [9]. One of the query expansion techniques is pseudo-relevance feedback that assumes that terms in retrieved documents are useful for expanding the original query. The criteria for selecting these terms can be based on a number of different features of documents and terms [9].

The idea of *monitoring* deep web content is discussed in several papers [19, 14, 1, 3, 6] which mostly focus on rerunning web data access approaches. Boncella *et al.* [6] provide an overview of how monitoring web data can be useful for competitive intelligence such as detecting cognitive hacking attacks on companies. Abiteboul [1] also discusses different aspects of monitoring both surface web and deep web data but without suggesting practical approaches. None of these studies focus on *efficiency* in monitoring change in deep web sources while highly limited resources of harvesting processes impose the need for efficient techniques.

Recent business applications for monitoring web data such as *BrightPlanet Deep Web Monitor*[1] (example of a deep web harvester) and *Google Aerts*[2] (example of a surface web monitoring tool) perform similarly based on a regular rerunning of their indexing or harvesting processes. *Google Alerts* alerts users for their topics of interest when new results for a given query are added to Google. Bharanipriya *et al.* [4] also discusses a number of harvesting tools that can monitor web pages by following the same principle of regular revisiting.

3. SOLUTION FOR WEB CONTENT MONITORING

As the first step to monitor web content changes, in focused web harvesting, we need a complete collection of relevant documents at hand. We use the introduced *Comb.LB-FB* method in [17, 16] to collect the first set of relevant documents. Comb.LB-FB is an efficient approach for focused web harvesting based on using a predetermined list of words and extracted content from previous search results.

After a predetermined time, set to two weeks in our experiments, we perform the second harvest. We re-harvest documents with the same queries to have a baseline for detecting changes. This second re-harvest is called *SecondCrawl* in our experiments. We define a changed document as a document with changed content compared to its previously retrieved version or a completely new retrieved document. Based on this definition, we detect and count the changed documents in SecondCrawl. We compare the content of two pages based on the discussed Shingle-Jacquard technique in Section 4.1.

In this work, our goal is to implement approaches that detect and harvest changed pages efficiently. Our proposed approaches are based on expanding a seed query (given entity). To expand a seed query, we favor terms that, in general, generate fewer duplicates and bigger samples and result in more changed and fewer unchanged pages. To find these terms, we focus on approaches inspired by pseudo-feedback methods for query expansion. In this method, the extracted content from previously retrieved documents is the source for a query generation process. Therefore, we employ different strategies such as analyzing the extracted content of returned results by a search engine and also lists of words from external corpora to expand a seed query. The proposed approaches are discussed in the following sections.

3.1 Most frequent from changed documents

In this method, the terms to form queries are selected from the content of previously retrieved pages. From these retrieved pages, the ones that are detected as changed pages are used for terms selection. Among the extracted terms from these changed documents, we select the terms with higher frequencies as they increase the chance of returning more results and therefore, bigger samples.

This method submits a seed query to a test search engine and detects the changed documents. The most frequent word in those changed documents is selected to be added to a seed query. The final step is submitting this formed query to a search engine. With the new results obtained from this query submission, these steps are repeated and new queries are formed and submitted. For example, for the given entity “PhD Comics”, the term “graduate” appears as the most frequent term in returned changed documents. The next step is submitting “PhD Comics”+“graduate”. This approach is called *MostFreq* method.

3.2 Least frequent from changed documents

The *LeastFreq* approach performs on the same basis of MostFreq but instead of selecting the most frequent terms, the least frequent ones are selected to reformulate queries. These least frequent terms have higher chances of reducing duplicates among changed documents.

3.3 Combined list and feedback from new documents

The most and the least frequent terms represent extreme cases of increasing chances of having either bigger samples or fewer duplicates. To reach a balanced approach, we need to determine a specific frequency between these two extreme

[1]https://www.brightplanet.com/solutions/deep-web-monitor
[2]https://support.google.com/alerts

cases for a term selection.

To do so, through Equation 1, with the size of a search engine $\texttt{size}^{\texttt{SE}}$ and the number of matching documents to a seed query $|R^{\texttt{Coll}}|$, we can calculate a frequency $|\texttt{Sample}^{Q}|$ to select terms from an external corpus. If the search engine size is unknown, we can estimate its size through approaches introduced by Khelghati *et al.* [15].

$$\frac{|R^{\texttt{Coll}} \cap \texttt{Sample}^{Q}|}{\texttt{size}^{\texttt{SE}}} = \frac{|R^{\texttt{Coll}}|}{\texttt{size}^{\texttt{SE}}} \times \frac{|\texttt{Sample}^{Q}|}{\texttt{size}^{\texttt{SE}}},$$
$$\texttt{where} \quad |R^{\texttt{Coll}} \cap \texttt{Sample}^{\texttt{Query}}| = l$$
$$\longrightarrow \quad |\texttt{Sample}^{Q}| = \frac{l \times \texttt{size}^{\texttt{SE}}}{|R^{\texttt{Coll}}|} \qquad (1)$$

The suggested approach calculates a specific frequency and creates a list of terms with this frequency from an external corpus (a list-based approach). Then, the terms from this list that also appear in the previously retrieved content (feedback-based approach) are selected to expand a given seed query [17].

To apply this approach to retrieve changed documents, instead of selecting terms from the previously retrieved content, a term is used for query reformulation if it appears in the changed content and it is also present in the extracted list from an external corpus. This technique is called *Combined* approach.

3.4 FedWeb

In *FedWeb* method, we analyze different versions of crawls of the web collected in different time points. The goal of this analysis is to detect changed documents and find the list of most representative words of this set of changed documents. We analyze two different versions of FedWeb collection to find a list of words that are more common in changed documents than unchanged ones. To expand a seed query in FedWeb approach, we choose a term from a list of words that are representatives of the changed documents in FedWeb collection.

To find a list of the most representative terms for changed documents, we apply a *Transformed Weight-normalized Complement Naive-Bayes (TWCNB)* classifier [21], implemented by Mahout [12], that seeks to maximize term weights on the likelihood that they belong to one class and do not belong to other classes [21]. In this classifier, documents are represented as vectors. The set of these vectors is shown as $\vec{d} = (\vec{d_1}, ..., \vec{d_n})$. Each document vector $\vec{d_1}$ has a label y_i. For any document in this set, d_{ij} is the count of word i in document d_j. By defining a smoothing parameter α for all words in the vocabulary and applying TF-IDF transformation and L2 length normalization, we can assign each term with a corresponding weight as shown in the following steps [21].

As the fist step, TF and IDF transformations and L2 length normalization of $\vec{d}$ are calculated by applying formulas in Equation 2. In Equation 2, i and m are counters for terms and j and k represent document numbers. In this equation, we define δ_{ij} as 1 if word i occurs in document j and 0 otherwise.

$$\texttt{TF}(d_{ij}) = \log(d_{ij} + 1) \qquad (\texttt{TF transformation}) \qquad (2a)$$

$$\texttt{IDF}(d_{ij}) = \texttt{TF}(d_{ij})(\log \frac{n}{\sum_{k=1}^{n} \delta_{ik}}) \quad (\texttt{IDF transformation}) \qquad (2b)$$

$$\texttt{LN}(d_{ij}) = \frac{\texttt{IDF}(d_{ij})}{\sqrt{\sum_{m=1}^{t} (\texttt{IDF}(d_{mj}))^2}} (\texttt{Length normalization}) \qquad (2c)$$

As the second step, to train a classifier for $(\vec{d}, \vec{y})$, $\vec{y} = (y_1, ...,$
$y_n)$, we calculate the weight w_{ci} of term i for label c, as shown in Equation 3. In this equation, α is the smoothing parameter and is set as $\sum_{i=1}^{t} \alpha_i$ with $\alpha_i = 1$ for all words. $\hat{\theta}_{ci}$ is the estimated probability that term i occurs in class c and do not belong to other classes.

$$\hat{\theta}_{ci} = \frac{\sum_{j:y_j \neq c}^{n} d_{ij} + \alpha_i}{\sum_{j:y_j \neq c}^{n} \sum_{m=1}^{t} d_{mj} + \alpha} \qquad (3a)$$

$$w_{ci} = \log \hat{\theta}_{ci} \qquad (3b)$$

$$w_{ci} = \frac{w_{ci}}{\sum_i |w_{ci}|} \qquad (\texttt{Weight normalization}) \qquad (3c)$$

To train this classifier, we need documents with assigned labels. We define two changed and unchanged labels. We compare the content of documents through Shingle-Jacquard text comparison technique. Based on the results of these comparisons, documents are assigned with changed or unchanged labels. Our trained classifier had an accuracy of %86.91. When the training is done, the classifier calculates the list of representing terms for each category [21]. After calculating these weights for all the terms in documents that belong to the changed documents category, the terms are ordered based on their weights. The top terms of this list are used to form queries to retrieve changed documents in our experiments. This method is referred as *FedWeb* approach.

4. EXPERIMENTS AND RESULTS

In our experiments, we run the proposed approaches on a test search engine to detect changed documents based on our change definition.

4.1 Experiments settings

We test our approaches on a real search engine. In these experiments, Google is considered as our test search engine. We believe Google is one of the most representative collections of the web including a wide range of domains and many entities. Although we target Google, there is no limitation on applying these approaches on other websites as far as they provide keyword-search.

Entities test set In our experiments, around 30,000 queries were submitted to download information for four different entities (“VitolâĂİ, “Ed BrinksmaâĂİ, “PhD ComicsâĂİ, and “Fireworks DisasterâĂİ) for the first round. These entities represent diverse types; Company, Person, Topic and Event. In addition to difference in type, we cover entities with the estimated numbers of matching results ranging from 2×10^4

to 5×10^5. These numbers are based on Google search estimates of matching documents.

Evaluation metric To assess approaches for returning changed documents, we need a metric that considers the differences in numbers of matching documents and change rates of entities. To do so, in Equation 4, we average the percentage of changed documents that a given approach *Appr.* returns for all the entities through dividing the number of returned changed documents by that approach for an entity E, $\texttt{CDocs}_{\texttt{Appr.}_{E_i}}$, by the total number of found changed documents for that entity $\texttt{TotalCDocs}_{E_i}$. This division is repeated for all the entities in the test set and averaged.

$$\texttt{Evaluate(Appr.)} = 100 \times \frac{\sum_{i=1}^{|\texttt{TestSet}|} \frac{|\texttt{CDocs}_{\texttt{Appr.}_{E_i}}|}{|\texttt{TotalCDocs}_{E_i}|}}{|\texttt{TestSet}|} \quad (4)$$

Detection of changed content In our experiments, we define two documents as near-duplicates by computing a Jacquard coefficient with a preset threshold of 0.9. We fix $k = 3$ as the shingle size. Let $\texttt{Shingles}(\texttt{Doc}_A)$ be the set of shingles in Doc_A and $\texttt{Shingles}(\texttt{Doc}_B)$ the set of shingles in $\texttt{Doc}_B$. We compute Jacquard coefficient through Equation 5. We extract the texts from web pages with similar URLs, calculate shingle sets of each document and save the fingerprints of shingles. For each pair of documents, if the Jacquard coefficient does not exceed the predefined threshold, we consider the pages as changed documents.

$$\texttt{Jacquard\_Coefficient}(\texttt{Doc}_A, \texttt{Doc}_B) = \frac{|\texttt{Shingles}(\texttt{Doc}_A) \cap \texttt{Shingles}(\texttt{Doc}_B)|}{|\texttt{Shingles}(\texttt{Doc}_A) \cup \texttt{Shingles}(\texttt{Doc}_B)|} \quad (5)$$

4.2 Results

In this section, the results of applying the introduced approaches in Section 3 to the test entities (Section 4.1) are presented. To establish a comparison baseline, we send the exact same queries from the previous crawl after two weeks and refer to it as *SecondCrawl*. The idea is to see the amount of occurred changes on the web for the exact same information needs.

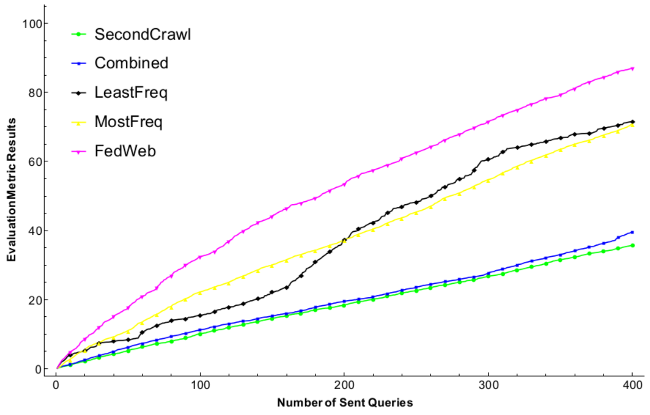

Figure 1: Performance of all the suggested approaches in returning changed documents for all test entities calculated through applying Equation 4

In Figure 1, the performances of all the introduced approaches in Section 3 in retrieving changed documents are compared. In this figure, for each submitted query, the defined evaluation metric in Equation 4 is calculated for all the approaches. This equation averages performances of an approach over all the entities for each query submission. For each query, the number of retrieved changed documents from all the previously submitted queries, up to and including that query, is divided by the total number of changed documents accumulated for a given seed query (test set entity) through all harvesting tasks. This is calculated for all the entities and averaged. This average is defined as the performance of an approach in retrieving changed documents for all the tested entities. As shown in Figure 1, FedWeb approach outperforms the other approaches by at least **20 percent** more changed documents retrieval. In following, we discuss the reasons behind this performance and mention our findings.

5. CONCLUSION AND FUTURE WORK

Our goal in this paper was to find efficient approaches to monitor web data for a topic of interest over time. In an ever-changing environment, devising efficient methods that can detect and retrieve changed web data for a given topic, from search engines imposing access limitations, is a challenging task. To address this challenge, we proposed four different approaches to efficiently harvest changed documents matching a given entity by querying a web search engine. Among these approaches, FedWeb approach outperformed the others. FedWeb, which is based on a set of representative terms of changed documents in FedWeb data collection, produced more unique changed documents while tested on the tested search engine and entities. This approach performed the best, with at least 20 percent difference from the other tested approaches, in returning more unique changed documents with the same number of queries.

In general, the results of this investigation show that monitoring web data in focused deep web harvesting can highly benefit from adapting query generation mechanisms to detect and retrieve changed content efficiently. We showed that instead of re-running harvesting processes, with a well-designed strategy, more changed documents can be retrieved with the same costs which is important in an environment of access limitations and limited resources.

Future work.

In this work, we investigated the effects of the frequency of words, presence of a word in retrieved documents and representative terms of changed documents in selecting the terms to form the best next query to submit. In addition to these factors, we consider a number of issues such as dependency of a query candidate to all previously submitted queries or further analysis of returned results by applying techniques such as entity extraction, entity disambiguation and text parsing, as important factors that need to be studied as future work to reach more efficient change detection techniques. As another future work, while we chose FedWeb collection which focuses on search engines in this work, we can consider ClueWeb or Common Crawl data collections. By analyzing two different versions of a large web crawl, we can either assign weights to websites representing their corresponding numbers of changed pages or train a classifier for terms and changed documents. Either through these weights or the classifier, we can select terms to expand a given seed query to form queries that can predict changes more accurately and efficiently.

Acknowledgement

This publication is supported by the Dutch national program COMMIT/.

6. REFERENCES

[1] Serge Abiteboul. Issues in monitoring web data. In Abdelkader Hameurlain, Rosine Cicchetti, and Roland TraunmÃijller, editors, *Database and Expert Systems Applications*, volume 2453 of *Lecture Notes in Computer Science*, pages 1–8. Springer Berlin Heidelberg, 2002.

[2] Manuel Álvarez, Juan Raposo, Alberto Pan, Fidel Cacheda, Fernando Bellas, and Víctor Carneiro. Deepbot: a focused crawler for accessing hidden web content. In *Proceedings of the 3rd international workshop on Data enginering issues in E-commerce and services: In conjunction with ACM Conference on Electronic Commerce (EC '07)*, DEECS '07, pages 18–25, New York, NY, USA, 2007. ACM.

[3] Michael K Bergman. White paper: the deep web: surfacing hidden value. *Journal of electronic publishing*, 7(1), 2001.

[4] V Bharanipriya and V Kamakshi Prasad. Web content mining tools: a comparative study. *International Journal of Information Technology and Knowledge Management*, 4(1):211–215, 2011.

[5] Krishna Bharat and Andrei Z. Broder. A technique for measuring the relative size and overlap of public web search engines. *Computer Networks*, 30(1-7):379–388, 1998.

[6] Robert Boncella. Competitive intelligence and the web. In *9th Americas Conference on Information Systems, AMCIS 2003, Tampa, FL, USA, August 4-6, 2003*, page 418. Association for Information Systems, 2003.

[7] Brian E. Brewington and George Cybenko. How dynamic is the web? In *Proceedings of the 9th International World Wide Web Conference on Computer Networks : The International Journal of Computer and Telecommunications Netowrking*, pages 257–276, Amsterdam, The Netherlands, 2000. North-Holland Publishing Co.

[8] Michael J. Cafarella. Extracting and querying a comprehensive web database. In *CIDR 2009, Fourth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 4-7, 2009, Online Proceedings*. www.cidrdb.org, 2009.

[9] Claudio Carpineto and Giovanni Romano. A survey of automatic query expansion in information retrieval. *ACM Comput. Surv.*, 44(1):1:1–1:50, January 2012.

[10] Carlos Castillo. Effective web crawling. In *ACM SIGIR Forum*, volume 39, pages 55–56. ACM, 2005.

[11] Junghoo Cho and Hector Garcia-Molina. The evolution of the web and implications for an incremental crawler. In *Proceedings of the 26th International Conference on Very Large Data Bases*, VLDB '00, pages 200–209, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[12] The Apache Software Foundation. *The Apache MahoutâĎć*, accessed Aug, 2015.

[13] Yeye He, Dong Xin, Venkatesh Ganti, Sriram Rajaraman, and Nirav Shah. Crawling deep web entity pages. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, pages 355–364, New York, NY, USA, 2013. ACM.

[14] Mohamamdreza Khelghati, Djoerd Hiemstra, and Maurice Van Keulen. Deep web entity monitoring. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13 Companion, pages 377–382, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee.

[15] Mohammadreza Khelghati, Djoerd Hiemstra, and Maurice van Keulen. Size estimation of non-cooperative data collections. IIWAS '12, pages 239–246, New York, NY, USA, 2012. ACM.

[16] Mohammadreza Khelghati, Djoerd Hiemstra, and Maurice van Keulen. Harvesting all matching information to a given query from a deep website, 2015.

[17] Mohammadreza Khelghati, Djoerd Hiemstra, and Maurice van Keulen. Towards complete coverage in focused web harvesting. In *Proceedings of the 17th International Conference on Information Integration and Web-based Applications & Services*, iiWAS '15, pages 65:1–65:9, New York, NY, USA, 2015. ACM.

[18] Steve Lawrence and C. Lee Giles. Accessibility of information on the web. *Nature*, 400:107–109, 1999.

[19] Bing Liu and Kevin Chen-Chuan Chang. Editorial: special issue on web content mining. *SIGKDD Explorations*, 6(2):1–4, 2004.

[20] Jayant Madhavan, David Ko, Lucja Kot, Vignesh Ganapathy, Alex Rasmussen, and Alon Halevy. Google's Deep Web crawl. *Proc. VLDB Endow.*, 1(2):1241–1252, August 2008.

[21] Jason D. M. Rennie, Lawrence Shih, Jaime Teevan, and David R. Karger. Tackling the poor assumptions of naive bayes text classifiers. In *In Proceedings of the Twentieth International Conference on Machine Learning*, pages 616–623, 2003.

[22] J. L. Wolf, M. S. Squillante, P. S. Yu, J. Sethuraman, and L. Ozsen. Optimal crawling strategies for web search engines. In *Proceedings of WWW '02*, pages 136–147, 2002.