Efficient Discovery of Authoritative Resources

Ravi Kumar #1, Kevin Lang #1, Cameron Marlow #2, Andrew Tomkins #1

#*Yahoo! Research, 701 First Ave., Sunnyvale, CA 94089, USA*

1{ravikumar,langk,atomkins}@yahoo-inc.com

2cameron@media.mit.edu

***Abstract*— Given a dynamic corpus whose content and attention are changing on a daily basis, is it possible to collect and maintain the high-quality resources with a minimal investment? We address two problems that arise from this question for hyperlinked corpora such as web pages or blogs: how to efficiently discover the correct set of authoritative resources given a fixed network, and how to track these resources over time as new entrants arrive, old standbys depart, and existing participants change roles.**

I. Introduction

We study the feasibility of gathering and maintaining highly authoritative content, at a scale several orders of magnitude smaller than the entire corpus. This approach is especially appropriate when the long tail of little-accessed content does not provide sufficient value to justify gathering and maintaining it.

The abstraction we consider is the following: given a dynamic corpus where content and attention is changing on a daily basis, is it possible to collect and maintain the high-quality resources with a minimal investment? We address two problems arising from this question, in the context of a hyperlinked corpus such as web pages or blogs.

Static: How to efficiently discover the correct set of authoritative resources given a fixed network?

Dynamic: How to track authoritative resources over time as new entrants arrive, old standbys depart, and existing participants change roles?

More specifically, we focus on a directed graph setting, where the resources are described by nodes and hyperlinks by edges. We consider various models of accessing this graph, namely, sampling nodes at random, performing a random walk, or crawling. Our measurement of authoritativeness is indegree. In the static setting, we propose a spectrum of algorithms, ranging from idealistic to realistic. We conduct extensive experiments on a blog data set and show that our realistic algorithms are very effective in identifying authoritative sources. Furthermore, they are practical and are highly competitive with their idealistic counterparts in terms of performance. Next, we turn to the dynamic setting, where the graph changes over time. In this setting we investigate the algorithms from the static case, augmented with several recrawl policies. Our experiments on another blog data set continuously crawled over time once again shows that it is possible to devise practical algorithms that are effective in tracking authoritative nodes in this graph.

Related work. Web crawling is a well-studied topic, and can be roughly subdivided into the problems of the *discovery* of new content [1] and *refresh* of updated content from previously visited resources. In the process of exploring the web, different strategies have been applied to the problem of ordering unvisited content; some have focused on the relative importance of topics [2], [3] while others have used the perceived rank [4], [5]. To effectively refresh updated pages some have modeled the change rates of individual pages on the web [6], [7], [8] while others have seen this as a problem of keeping a given set of content up-to-date [9], [10], [11], [12]. These approaches aim to characterize the web at large, attending to comprehensiveness and coverage; we, instead, focus on finding and tracking the most important content.

The process of identifying top resources in a web context is also analogous to the task of identifying and threading together important topics in a stream of information. Much attention has been paid to this problem through the iterative improvements on a fixed corpora on two sub-problems: *topic detection*, or the identification of emerging, related concepts and *event tracking*, or tracking subsequent events given an initial set [13], [14]. Similar efforts have also been made to identify bursty events in streams of data [15].

II. Discovery in static graphs

In this section we discuss the basic problem of discovering authoritative nodes in a fixed graph. We present several algorithms for this problem and analyze the performance of our algorithms on a large blog data set.

Problem. Let $G = (V, E)$ be a directed graph with node set V and edge set $E = \{(u, v) \mid u, v \in V\}$. The *inlinks* of a node u are the edges $\{(v, u) \mid (v, u) \in E\}$; the size of this set is the *indegree* of u, denoted $\mathrm{id}(u)$. The *outlinks* of a node u are the edges $\{(u, v) \mid (u, v) \in E\}$; the size of this set is the *outdegree* of u, denoted $\mathrm{od}(u)$. We will define a node to be "authoritative" if it has a high indegree; we note that indegree is easy to interpret and compute, and it cannot be easily changed by the node itself.

The problem of finding high-quality nodes in a graph can be stated as follows: given a directed graph G and a target number k, find the k most authoritative nodes. Exactly finding that subset K will be difficult, so instead we ask for an approximate subset $C \subset V, k < |C| \ll |V|$ to maximize the quantity $|C \cap K|/|K|$, called the *crawl performance*. A stricter measure that might be more relevant in real applications is the *rank performance*, which is given by $|K_H \cap K|/|K|$, where $K_H \subseteq C$ with $|K_H| = k$ is the output of the algorithm.

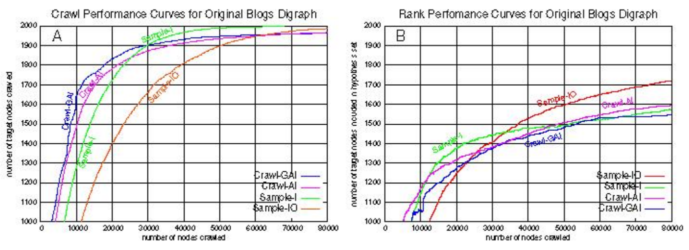

Fig. 1. Crawl and rank performance curves

Several of our algorithms will maintain a set C of crawled nodes that we hope will contain many nodes from K. For crawl-based algorithms, a subset of nodes of special interest is the *frontier* $F \subseteq V$ of the crawl. These are nodes that are one outlink away from the crawled set but not themselves crawled yet. While the algorithm does not know the outdegree of a node $u \in F$ in the frontier, it can obtain the apparent indegree of u based on its knowledge of the graph so far; we denote this by $\tilde{\mathrm{id}}(u)$.

Algorithms. We first present two sampling-based algorithms, which are unrealistic since they assume complete global knowledge of the degrees of the nodes in G. The algorithms Sample-I, Sample-IO correspond to random sampling with probability of choosing u proportional to the indegree $\mathrm{id}(u)$, and the sum $\mathrm{id}(u) + \mathrm{od}(u)$, respectively. Next we present two realistic crawling algorithms which maintain a set of the nodes that have already been crawled along with a frontier for exploration. They also maintain apparent indegrees for nodes in the frontier. At each step the algorithm must decide which node in the frontier must be crawled next. In Crawl-AI the next node to crawl is a random frontier node, with probability proportional to apparent indegree. In the greedier Crawl-GAI method, the next node to crawl is the frontier node $u \in F$ with the highest apparent indegree $\tilde{\mathrm{id}}(u)$.

Experimental results. Our main data source comes from a weblog corpus collected from May 16th to June 21st, 2005 for the MIT Weblog Survey [16]. These blogs were discovered using the Blo.gs ping service that blog software uses to automatically notify subscribers when a given weblog is updated. Over the course of the 37 day period, 15 million links were extracted from about 1 million observed weblogs. These data were cleaned of outliers and anomalies according to [16]. Subsequently, the graph contains 343,743 nodes and 1,571,772 directed edges. For all of the crawl-based algorithms, we used a fixed arbitrarily chosen source node u_0 that belongs to the graph's SCC.

In Figure 1-A we plot the *crawl performance* of several algorithms. We have defined the target set K to be the 2000 nodes with the highest indegree. The plot's x-axis is the number of nodes crawled so far, while the y-axis is the number of nodes in K that have been visited. Sample-I works better than the other sampling algorithms (mostly not shown) which is not surprising since it is directly sampling on indegree (our measure of authoritativeness). Interestingly, Crawl-GAI and Crawl-AI find high-indegree nodes even faster than Sample-I, at least at the beginning.

As one might expect, scores for the stricter *rank performance* measure (Figure 1-B) are lower than crawl performance scores. (Figure 1-A). Also, the relative order of the algorithms has nearly reversed. In particularly, the greedy Crawl-GAI algorithm is the worst for this measure. Apparently pulling high indegree nodes into the pool too rapidly causes apparent indegree to become a bad approximation to true indegree. There is an exploration-exploitation tradeoff here that will be discussed more in the next section. [Also, we note that the longer version of this paper contains more algorithms and many more plots].

III. Tracking in dynamic graphs

In this section we expand our scope to include *time graphs* [17], in which each edge arrives at a particular point in time. We will define a time-varying target set of popular nodes that our algorithms must track.

Problem. A time graph $G = (V, E)$ is a set of nodes V, with a set of timestamped directed edges E, where $(u, v, t) \in E$ represents an edge from u to v arriving at time t. Let w be a fixed time window. The induced subgraph G_{t_0} at time t_0 is defined as $G_{t_0} = (V, E_{t_0})$, where $E_{t_0} = \{(u, v, t \in E) \mid t \in [t_0 - w, t_0]\}$. The outlinks of a node u at time t_0 are given by $\mathrm{out}(u, t_0) = \{(u, v, t) \in E_{t_0}\}$. The target k nodes to track at time t, denoted K_t, are simply the top k nodes of G_t ranked by decreasing indegree. We use $w = 7$ days and $k = 500$ in our experiments.

When an algorithm performs a crawl of node u at time t_0, it sees $\mathrm{out}(u, t_0)$ as a result. At this time, the algorithm is said to have crawled u. Let C_t be the set of all nodes crawled by time t and F_t be the frontier at time t. As before, our primary figure of merit is crawl performance, defined to be $|C_t \cap K_t|/|K_t|$ at time t. In some cases, we may also study the *discovery performance* at time t, which is given by $|(C_t \cup F_t) \cap K_t|/|K_t|$. This measure shows how effectively the algorithm has learned about target nodes, even if it has not yet selected them for crawling.

Algorithms. Dynamic algorithms, in addition to deciding which node to explore next, must also interleave recrawls of already-crawled nodes in order to harvest newly-arriving links. We distinguish between an *expansion procedure*, as employed by a static algorithm, and a *recrawl procedure* for selecting already-crawled nodes to revisit. We assume that a dynamic algorithm will interleave calls to these two procedures at random with some probability fixed in advance (taken to be 0.5 in our experiments).

The *yield* of a recrawl event is define as the number of links that were not present during the prior crawl of that node, divided by the time since the prior crawl. Each node that has been crawled at least twice has a yield score defined by the yield of the most recent crawl. *Link score* is defined analogously to yield, but takes into account the perceived indegree of the new links on a node: the link score of a

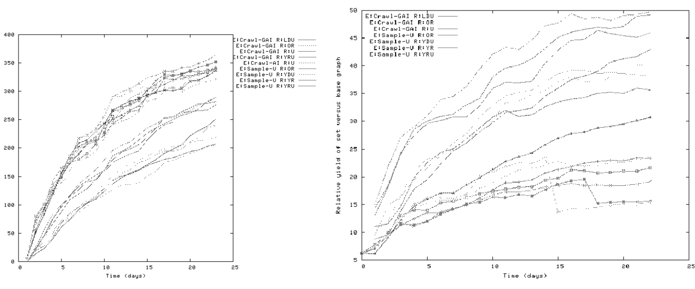

Fig. 2. Crawl/discovery performances and yield.

recrawl of a node is the sum over each outlink not present during the prior crawl of the number of inlinks seen so far to the destination. Thus, link score prefers nodes that discover "high-quality" nodes, according to indegree. The link score of a node is then the link score of the most recent recrawl.

Section II introduced a variety of expansion procedures of which we will study the following three: omniscient algorithm Sample-U and realistic algorithms Crawl-GAI, Crawl-AI. We introduce the following recrawl policies. In policy U, we pick a node for recrawl uniformly at random and in policy OR, we pick a node with probability proportional to the total outlinks seen at this node to date. In policy YR, we select the node at random with probability proportional to yield score. Policy YRU implements a uniform mixture of YR and U; this helps to avoid the bootstrapping problem — to estimate yield, a node must be crawled at least twice. Policy LDU is a uniform mixture of policy U and the policy of picking the node with highest link score.

Experimental results. We report on a series of experiments on dynamic data. The data used for this analysis is made available as part of the first ICWSM Conference on Weblogs and Social Media, and consists of about 14M posts from 3M weblogs, representing about 10G compressed. We extracted all links from the data, resulting in 340K links over a period of 24 days. Because the content consists just of blog entries, this data does not contain any template material, blogrolls, or other links beyond what occurs in the entry itself. Thus, the average degree in the subgraph induced by weblogs in the crawl is quite low, but the link quality is high. Each link is annotated with a timestamp given in seconds.

Figure 2-A shows the results for crawl performance with regular lines, and the results for discovery performance are also shown, using the same colors, with hash marks added. The Crawl-GAI expansion algorithm with the OR recrawl algorithm performs well in both cases. Surprisingly, the more sophisticated algorithms based on yield and link score do not perform better.

Figure 2-B shows the effectiveness of the algorithm at identifying target nodes, compared to selecting random nodes from the graph. The smooth curves show the probability that a uniformly-chosen crawled node is a target node, divided by the probability that a uniformly-chosen node from the graph is a target node. The hashed lines show the probability that a uniformly-chosen node from $C_t \cup F_t$ is a target node, divided by the probability that a uniformly-chosen node from the graph is a target node. The union of crawled nodes and frontier nodes are more than 20 times more likely to be target nodes than random nodes from the graph, and the ratio for crawled nodes is even higher, reaching almost 50 for the best algorithm, which is a realistic algorithm (expand using Crawl-GAI, recrawl using OR). The decline in quality of the frontier for certain algorithms comes about late in the crawl because nodes with large numbers of outlinks to new but low-quality content are discovered. We may also observe from this graph that the algorithm effectively learns from its experiences, doing a better job of distinguishing high-quality nodes from low-quality nodes as the graph grows.

IV. Conclusions

We presented a problem exposing the tradeoff between coverage and resources in the case of hyperlinked media. We evaluated the performance of many expansion techniques in a static graph and recrawl methods in a dynamic graph. Our best algorithms — Crawl-GAI and Crawl-AI for expansion and U and YDU for recrawl — performed surprisingly well, often beating those that had an omniscient advantage.

References

[1] A. Dasgupta, A. Ghosh, R. Kumar, C. Olston, S. Pandey, and A. Tomkins, "The discoverability of the Web," in *Proc. 16th WWW*, 2007, pp. 421–430.
[2] S. Chakrabarti, M. van den Berg, and B. Dom, "Focused crawling: a new approach to topic-specific web resource discovery," *Computer Networks*, vol. 31, pp. 1623–1640, 1999.
[3] B. Pinkerton, "Finding what people want: Experiences with the webcrawler," in *Proc. 1st WWW*, 1994.
[4] S. Abiteboul, M. Preda, and G. Cobena, "Adaptive on-line page importance computation," in *Proc. 12th WWW*, 2003, pp. 280–290.
[5] P. Boldi, M. Santini, and S. Vigna, "Do your worst to make the best: paradoxical effects in pagerank incremental computations," in *Proc. 3rd WAW*, 2004, pp. 168–180.
[6] B. Brewington, G. Cybenko, R. Stata, K. Bharat, and F. Maghoul, "How dynamic is the web?" in *Proc. 9th WWW*, 2000, pp. 257–276.
[7] J. Cho and H. Garcia-Molina, "The evolution of the web and implications for an incremental crawler," in *Proc. VLDB*, 2000, pp. 200–209.
[8] D. Fetterly, M. Manasse, M. Najork, and J. Wiener, "A large-scale study of the evolution of web pages," *Software Practice and Experience*, vol. 34, no. 2, pp. 213–237, 2004.
[9] J. Cho and H. Garcia-Molina, "Synchronizing a database to improve freshness," *SIGMOD Record*, vol. 29, no. 2, pp. 117–128, 2000.
[10] J. E. Coffman, Z. Liu, and R. R. Weber, "Optimal robot scheduling," *Journal of Scheduling*, vol. 1, no. 1, 1998.
[11] S. Pandey and C. Olston, "User-centric web crawling," in *Proc. 14th WWW*, 2005, pp. 401–411.
[12] J. L. Wolf, M. S. Squillante, P. S. Yu, J. Sethuraman, and L. Ozsen, "Optimal crawling strategies for web search engines," in *Proc. 11th WWW*, 2002, pp. 136–147.
[13] J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang, "Topic detection and tracking pilot study: Final report," in *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, 1998. [Online]. Available: citeseer.ist.psu.edu/allan98topic.html
[14] J. Allan, R. Papka, and V. Lavrenko, "On-line new event detection and tracking," in *Proc. 21st SIGIR*, 1998, pp. 37–45.
[15] J. Kleinberg, "Bursty and hierarchical structure in streams," *DMKD*, vol. 7, no. 4, pp. 373–397, 2003.
[16] C. Marlow, "The stuctural determinants of media contagion," Ph.D. dissertation, Massachusetts Institute of Technology, 2005.
[17] R. Kumar, J. Novak, P. Raghavan, and A. Tomkins, "On the bursty evolution of blogspace," *World Wide Web Journal*, vol. 8, no. 2, pp. 159–178, 2005.