Analyzing fine-grained hypertext features for enhanced crawling and topic distillation

Soumen Chakrabarti
Indian Institute of Technology Bombay *
Ravindra Jaju
Indian Institute of Technology Bombay
Mukul Joshi
Indian Institute of Technology Bombay
Kunal Punera
Indian Institute of Technology Bombay

Abstract

Early Web search engines closely resembled Information Retrieval (IR) systems which had matured over several decades. Around 1996–1999, it became clear that the spontaneous formation of hyperlink communities in the Web graph had much to offer to Web search, leading to a flurry of research on hyperlink-based ranking of query responses. In this paper we show that, over and above inter-page hyperlinks, much semantic information can be teased out of the manner in which markup tags, such as menu-bars, tables, and lists are used to organize pages, and the context in which hyperlinks are made from a page to another. We believe that delving into fine-grained page structure will be the next wave in hypertext mining, bridging some of the gap between unstructured HTML and relatively structured XML, and preparing the stage for deeper analysis into page snippets. We talk about two applications to illustrate fine-grained page analysis: topic-guided focused crawling, and hyperlink-based ranking which takes page structure into account. In both cases, we show that exploiting fine-grained structure enhances the performance of our systems.

1 Introduction

Traditional IR has considered the *document* as the fundamental unit of analysis [18, 17]. The application of IR to Web search has continued this view, regarding whole *Web pages* as documents. Crawlers fetch content in units of whole pages, and rarely discriminate between outlinks on semantic grounds when they scan a page for new URLs to be crawled. Search engines return ranked lists of whole pages as well. Recent link analysis algorithms which rank pages on the basis of the prestige bestowed by inlinks [2, 11] (a technique called *topic distillation*) also model the Web graph at a coarse grain, with whole pages as single nodes. A score is attributed to each page and links from one page to another are viewed as a recommendation of all of the target page's content by the link creator. This *coarse-grained* approach to crawling and ranking betrays an assumption that Web pages are homogeneous in their contents, and all outlinks are equally related to the contents.

We believe that a great deal of semantic information, latent in the HTML sub-structure, is lost when one considers Web pages as indivisible. HTML tags and their arrangement as the Document Object Model or DOM tree (`http://www.w3.org/DOM/`) provide clues which prove valuable for crawling and IR tasks. Therefore, we enhance the coarse-grained graph model for the Web to a **fine-grained model** in which every

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

*Contact author, email `soumen@cse.iitb.ac.in`

page is represented by a DOM tree, with edges internal to a DOM tree representing local tag-tree structure and ordinary hyperlinks as edges connecting two DOM trees.

At the Laboratory for Intelligent Internet Research[1] in IIT Bombay, we are using this fine-grained model for improving a number of Web IR and mining tasks. In this paper we concentrate on two applications: *focused crawling* [6] and *topic distillation* [3, 5]. We intend to make our prototypes (HTML parser and cleaner, crawler, and distillation programs) available in the public domain for research use.

The rest of the paper is arranged as follows. Section 2 describes our data model and the various issues in the parsing and sanitization of HTML pages. In the next two sections we outline two applications of the fine-grained model. Section 3 describes the how the new model can be used to improve the rate at which a focused crawler acquires relevant pages. In Section 4 we show that the application of the fine-grained model to ranking Web pages improves full-page approaches. We conclude with comments on ongoing and future work in Section 5.

2 HTML tags and parsing

In a perfectly engineered world, Web pages will be written in a markup language more sophisticated than HTML, with a universally accepted meta-data description format. XML (`http://www.w3.org/XML/`) and RDF (`http://www.w3.org/RDF/`) have made great strides in that direction, especially in specific domains like e-commerce catalogs, electronic components, genomics, and molecular chemistry. However, we believe that the Web will always remain adventurous enough that its most interesting and timely content can never be shoehorned into rigid schemata. Without downplaying the importance of XML and associated standards, we emphasize the need for HTML to DOM converters which capture the fine-grained Web page model accurately.

HTML is primarily used to specify the visual formatting of the content of a Web page, but some Web authoring idioms and the organizational functions of certain tags also help us derive semantic information. Creators of links between Web pages normally provide some information about the target page around the URL on the source page, not necessarily in the anchor text alone. E.g., a section may start with "`<h1>Japanese car makers</h1>`" and continue with a list of links, including `http://www.honda.com/` and `http://www.mazda.com/`, sites which may choose not to describe themselves in those words. Furthermore, the occurrence of two such links at close proximity (or inside, say, a `<td></td>` tag) might suggest that they point to pages similar in content. Conversely, while crawling for pages about mountain biking, we would do well to determine the DOM subtrees which are semantically dissimilar and avoid exploring links emerging from irrelevant subtrees.

The DOM tree of a page can be obtained by parsing the HTML. This, however, is not as easy as it seems. A large fraction of HTML pages on the Web violates the HTML standard. Patching the syntax is non-trivial because we wish to retain as much DOM information as we can for later analyses. For example, a `<tr>` tag implies that a `<table>` tag must have already occurred somewhere prior to it. If an enclosing `<table>` tag does not exist, it has to be inserted, but we must make sure that such an insertion does not violate other HTML syntax rules. An actual HTML page had a `<form>` tag starting outside a `<table>`, and closing within it. This caused an early version of the cleaner to close the `<table>` tag before the `</form>` tag, leading to loss in both formatting and information.

No movement of text or tags is done in the cleaning phase itself because not only can it change, albeit unintentionally, what the original author probably meant to convey, but it also makes the code increasingly complex and consequently prone to errors. Movement, especially of empty tags which might be created by the cleaner due to inappropriate use of tags, is attempted in the DOM tree construction phase using a variety of heuristics. The default patching policies can be modified by the user through a simple API.

[1] `http://www.cse.iitb.ac.in/laiir/`

3 Focused crawling guided by DOM structure

The Web has over two billion static pages today and an estimated 600 gigabytes of text changes per month [10]. General-purpose crawlers dropped to a mere 18% coverage of the Web in 1999 [13]. Matters have improved since, Google now covering over half the estimated extent of the Web. However, given the size of the Web, coverage often comes at the price of freshness, large crawlers visiting most sites every several weeks at best.

In contrast to general purpose crawlers, a **focused crawler** seeks to collect pages related to specific topics, which it learns from examples provided by the administrator. A federation of focused crawlers can cover each specific topic in more depth, and succeed in keeping crawls fresher. They start from example URLs on a certain *focus topic*, say, basketball, and explore outlinks with the goal of collecting pages about basketball and avoiding pages that are not about this topic.

We initiated the design and study of focused crawlers in 1999 [7], and several enhancements have been suggested since then [9, 15]. The key operation of a focused crawler is to estimate the worth of fetching a new page v based on its citation found on a page u that has already been fetched. Such decisions are made by training a *text classifier* [14] on the example URLs.

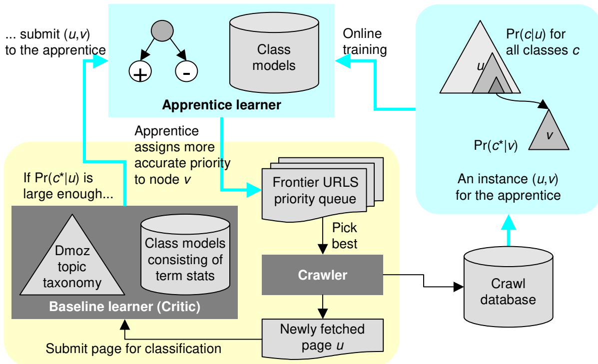
3.1 Fine-grained focused crawling

Under the coarse-grained model, a focused crawling system can avoid fetching irrelevant pages (pages not about the focus topic) by not crawling links that it finds on irrelevant pages. For example, the relevance of u to focus topic c_* (which, in a Bayesian setting we can denote by $\Pr(c_*|u)$) could be used as an estimate of $\Pr(c_*|v)$, the relevance of unvisited outlink to page v. The assumption made here is that if a page is not relevant to the focus topic, the chances that links found on that page point to relevant resources are remote (in other words, that pages across a hyperlink are more similar than two randomly chosen pages on the Web). For obtaining $\Pr(c_*|u)$ we use the RAINBOW naive Bayes classifier by McCallum and others [14]. The classifier is trained offline on examples taken from a taxonomy such as Yahoo! (`http://www.yahoo.com/`) or the DMoz Open Directory (`http://dmoz.org/`).

Consider, however, a Web page u which talks about various topics in general, but has a few links to very good resources/pages (v_i) on basketball. Because u relates to sundry topics, it may not be considered relevant by a focused crawler crawling pages on basketball, and the links to basketball resources (v_i) will remain unexplored. Conversely, a page may contain extremely relevant text but some links to irrelevant pages (see the examples in Figure 8). These irrelevant links would be eagerly crawled because they were found on a highly relevant page. Clearly, the judgments on relevance of v_i ($\Pr(c_*|v_i)$) must be based on something more than just $\Pr(c_*|u)$.

In the fine-grained focused crawler [6] we use **two** classifiers. In a standard focused crawler, the classifier was used to assign priorities to unvisited frontier nodes v. This no longer remains its function. The role of assigning priorities to unvisited URLs in the crawl frontier is now assigned to a new learner called the *apprentice*, and the priority of v is specific to the (u, v) link which leads to it. Meanwhile the role of the standard classifier becomes one of generating instances for the apprentice, as shown in Figure 1. We may therefore regard the baseline learner as a *critic* or a *trainer*, which provides feedback to the apprentice so that it can improve "on the job."

We use RAINBOW for the apprentice classifier as well, suitably encoding the context of each link (u, v) so that the bag-of-words classifier can make use of these encoded features effectively. After parsing the HTML and creating a DOM tree, its leaves are numbered from left to right. The specific `<a href...>` which links to v is an internal node a_v, and is the root of the subtree which contains the anchor text of (u, v). The subtree may contain other nodes representing tags such as `<em>` or `<b>`. We regard all textual tokens found in this subtree to be *at offset zero* w.r.t. the (u, v) link. Tokens outside the subtree of a_v are associated with negative and positive offsets as shown in Figure 2. For efficiency, we limit the maximum offset up to which we accept tokens to some **window** around a_v of radius $d_{\max}$. Special strings can now be prefixed to the text tokens to encode their location



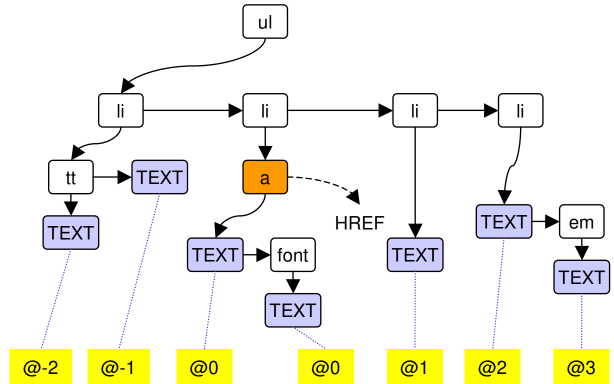
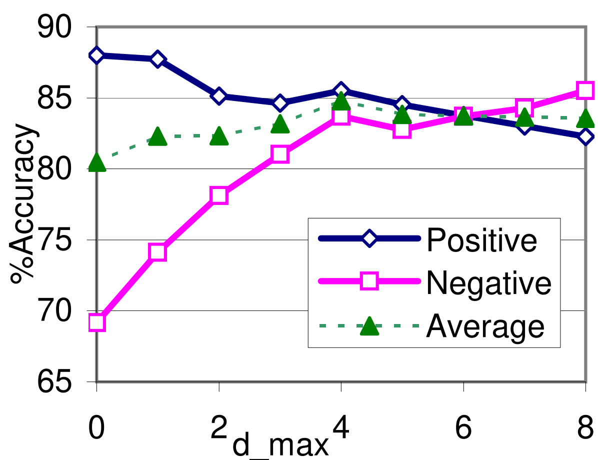


Figure 1: The apprentice is continually presented with training *cases* (u, v) with suitable features. Once trained, the apprentice helps assign unvisited nodes v better estimates of relevance.

Figure 2: Numbering of DOM leaves used to derive offset attributes for textual tokens. '@' means "is at offset".



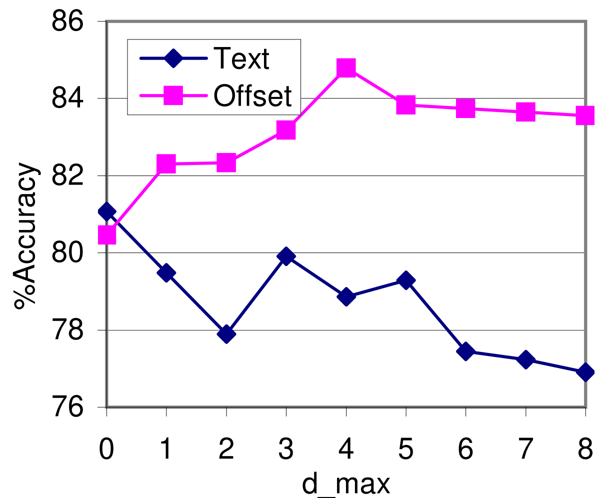


Figure 3: Apprentice accuracy visibly improves as $d_{\max}$ is increased, up to about 5–7, mainly because of an increased ability to correctly reject negative (low relevance) outlink instances.

Figure 4: Using encoded offset as against plain text improves the apprentice's accuracy.

w.r.t. to the (u, v) link.

3.2 Experiments and Results

We experimented with focused crawls starting with examples of dozens of topics from roughly 450 topics chosen from DMoz. We excerpt some salient observations.

Window width: A key concern in the selection of the context of a link is the window size. In the typical example shown in Figure 3, we see that the accuracy of the apprentice classifier increases rapidly as $d_{\max}$ is increased above 0 (indicating that using only anchor text loses out on useful features) but the gains level out and may even fall for larger $d_{\max}$. Based on these observations we settled on a maximum window size of 5.

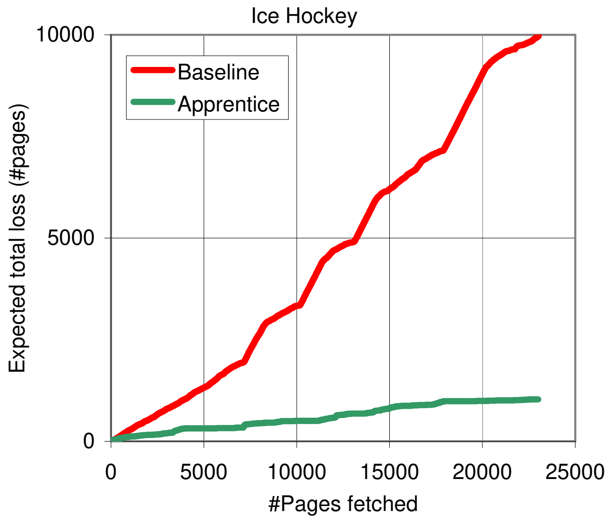


Figure 5: Guidance from the apprentice significantly reduces the loss rate of the focused crawler. In this chart the apprentice has been trained offline.

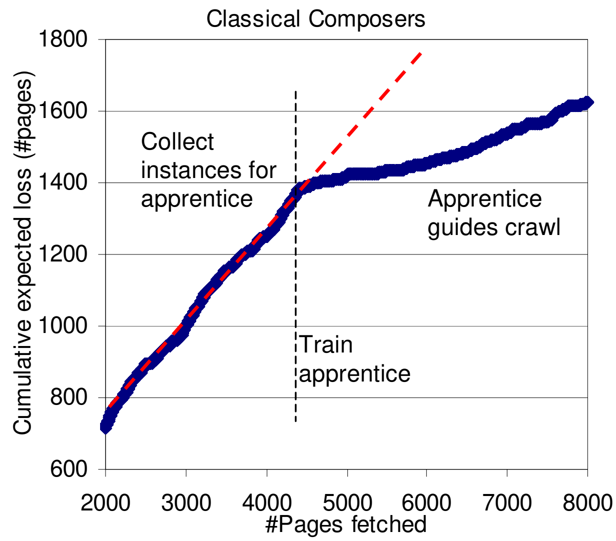


Figure 6: An example of training the apprentice *on-line* followed by using it for crawl guidance. Before guidance, loss accumulation rate is over 30%, after, it drops to only 6%.

Encoding DOM offsets in features: We verified that adding offset information to text tokens was better than simply using plain text near the link [4]. One sample result is shown in Figure 4. The apprentice accuracy decreases with $d_{\max}$ if only text is used, whereas it increases if offset information is provided. This highlights the importance of proper *feature extraction*.

Rate of acquiring relevant pages: The focused crawler's performance can be measured in terms to its *harvest rate* which is the fraction of relevant pages collected. If V is the set of nodes collected, the harvest rate is defined as $(1/|V|)\sum_{v \in V} \Pr(c_*|v)$. Alternatively, we can measure the *loss rate*, which is one minus the harvest rate, i.e., the (expected) fraction of fetched pages that must be thrown away. We present loss rates, because this is the part of the crawler's work which can be potentially avoided. Figure 5 contrasts the loss rate of the baseline and apprentice guided crawls. Figure 6 shows the sudden decrease in loss rate when the focused crawler switches from using the baseline classifier to the apprentice classifier once the latter has been sufficiently trained.

4 Topic Distillation

Once a focused crawler has collected a substantial corpus of the Web pages on a specified topic, we are usually interested in compiling a list of authoritative pages and a list of link collections pertaining to that topic [12]. Such lists can be computed using **topic distillation**, a process of discovering *authoritative* Web pages and comprehensive collections of links called as *hubs* which reciprocally endorse each other and are relevant to a given topic. Kleinberg's HITS [11] and the PageRank algorithm [2] underlying Google are the most well known topic distillation algorithms. In this paper we concentrate on HITS.

A key property of HITS is the diffusion of endorsement or popularity across cocited siblings. From the equations shown in the Figure 7 we notice that the increase in the authority score of v_1 gives rise to the increase in authority score of v_3 within the next two iterations. We call this phenomenon *authority diffusion*.

Because HITS models the hub page u as a single node with a single hub score, high authority scores could diffuse from a relevant authority v_1 to a less relevant cocited authority v_3. This makes HITS and related algo-

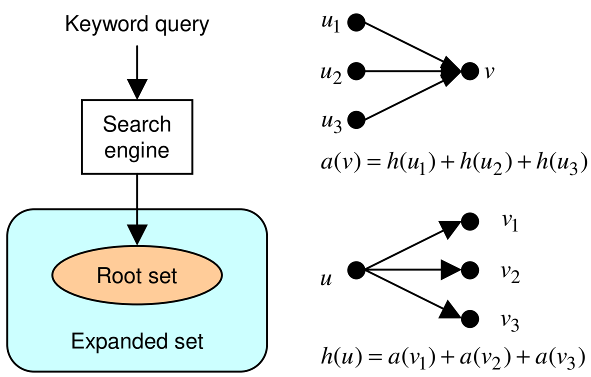


Figure 7: Hyperlink induced topic search (HITS).

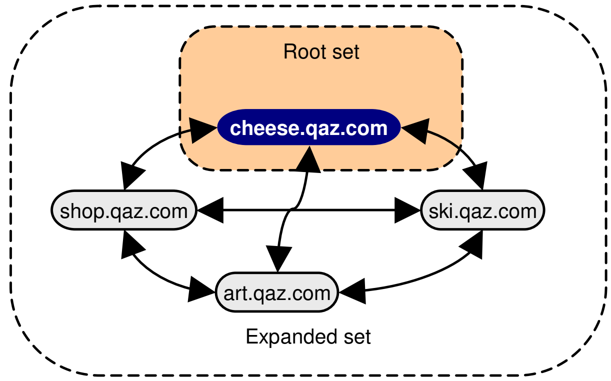


Figure 8: Spammed or mixed hubs can confound topic distillation.

rithms vulnerable to unintended *topic drift* and deliberate *clique attacks*. Topic drift may happen if v_1 is a studio homepage, v_3 is a page about Oscar nominations, and only one topic is of interest to the user. Clique attacks are perpetrated by site designers aiming to improve their clients' search engine ratings by artificially constructing dense hyperlink neighborhoods that appear like prestige-conferring citations (Figure 8). These links imply no semantic coherence or editorial judgment, which are crucial for the success of topic distillation. Banners, navigational panels, advertisements, and "Web ring" links add to the woes of traditional distillation algorithms.

4.1 Fine-grained topic distillation

Our enhancement to the standard topic distillation algorithm is largely contained in one step: the authority-to-hub score transfer. Recognizing that a single hub score per page may result in unwanted score diffusion, we represent each page using its DOM tree. To simplify the description for this paper, `<a href...>` nodes (Figure 4) are made *leaf* nodes, removing their subtrees. These leaves are called **microhubs**.

HITS in effect aggregates all microhub scores on a page and makes the sum the hub score for the whole page. We would like to aggregate microhub scores selectively. If we believe that some scores belong to relevant DOM subtrees, we would like to add them up and propagate them back to the authority nodes as in HITS. But we would *not* like to aggregate over microhub scores which we believe lead to irrelevant authorities.

There are (at least) two kinds of clues as to whether a DOM subtree is relevant. First, if a disproportionately large fraction (compared to the global average) of outlinks in a DOM subtree link to known good authorities, we should collapse the hub score of the subtree. Second, if the text contained in a subtree is highly relevant to the query (compared to neighboring subtrees), that subtree could represent a single hub.

Informally, our model of hub generation enables our algorithm to find a *frontier* across each hub's DOM tree. Subtrees attached to the frontier are made individual nodes in the distillation graph. Thus the hub score of the entire page is *dis-aggregated* at this intermediate level. The key question is: how do we determine such a frontier for each page? Owing to space limitations, we will take up this question with regard to the first clue above: the proximity of outlinks to good authorities.

Finding the best frontier: Our goal is to recover the *simplest* frontier which best explains any non-uniformity in the microhub scores. We wish to avoid choosing either one large and redundant frontier near the leaves and explain the many small and homogeneous subtrees very accurately, or choosing a short frontier near the root with a few large subtrees which are harder to model since they contain many diverse microhub scores.

```
h ← Ea
for each document DOM root u do
  frontier F ← segment(u)
  for each frontier node v ∈ F do
    h(v) ← ∑_{w∈L_v} h(w)
    for each w ∈ L_v do
      h(w) ← h(v)
    end for
  end for
end for
a ← E^T h
normalize a so that ∑_u a(u) = 1
```

Figure 9: One iteration of the fine-grained topic distillation algorithm. L_v is the leaf microhubs in the DOM subtree rooted at v.

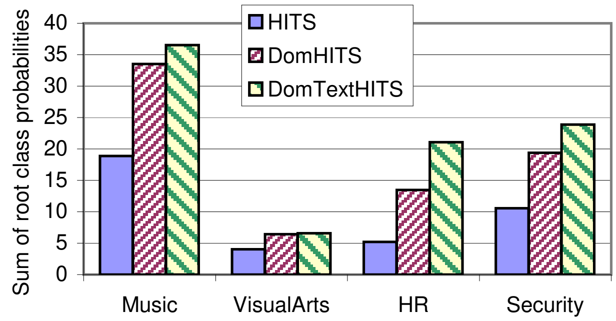


Figure 10: Authorities reported by DOMTEXTHITS have the highest probability of being relevant to the Dmoz topic whose samples were used as the root set, followed by DOMHITS and finally HITS. This means that topic drift is smallest in DOMTEXTHITS.

We strike a balance between these extreme choices by designing comparable units of cost for defining a frontier model (the **model cost**) and generating microhub scores from a frontier model (the **data cost**), and minimizing the sum of these two costs, following Rissanen's Minimum Description Length (MDL) principle [16].

For any node u, we can model the microhub scores H_u at the leaves of the tree rooted at u using a distribution Θ_u. The probability of a specific microhub score h w.r.t. this distribution is denoted $\Pr_{\Theta_u}(h)$. Let the global distribution at the root DOM node be Θ_0. Let the model cost Θ_u for a node u on the frontier, encoded relative to the global distribution Θ_0, be denoted $L(\Theta_u|\Theta_0)$. The data cost for the set H_u of microhub scores in the subtree rooted at u is approximately $-\sum_{h\in H_u}\log\Pr_{\Theta_u}(h)$ [8].

Finding the best frontier is NP-hard. The following greedy strategy worked quite well: at each node u, we compare two options:

- Make u a frontier node, paying $L(\Theta_u|\Theta_0)$ and $-\sum_{h\in H_u}\log\Pr_{\Theta_u}(h)$.
- Expand u to its children $\{v\}$, paying $\sum_v L(\Theta_v|\Theta_0)$ and $-\sum_v\sum_{h\in H_v}\log\Pr_{\Theta_v}(h)$.

We greedily pick the locally better option and continue until the frontier can no longer be improved by pushing it down. When a node is not worth expanding, we say it is **pruned**. The DOM segmentation steps above are integrated into the standard topic distillation algorithm by inserting a call to the **segment** routine after the $\mathbf{h} \leftarrow \mathbf{Ea}$ step and before the $\mathbf{a} \leftarrow \mathbf{E^T h}$ step (Figure 9).

We can use a similar segmentation technique for the text associated with DOM subtrees. The combined evidence from microhub scores and text can lead to a more reliable hub segmentation than using any one by itself. We call this algorithm as DOMTEXTHITS; the details of this algorithm have been reported elsewhere [5].

4.2 Experimental study

Our system can be used with ad-hoc queries as well as predefined topics; all we need is a subgraph of the Web with a significant fraction of the pages being relevant to a broad topic. We used the set of 28 queries used in the ARC/Clever system and other researchers [1, 4]. We also picked 20 topics from a simplified version of the DMoz topic hierarchy (`http://dmoz.org`). There were about 4 million non-local `HREF`s and over 15 million fine-grained nodes and links in our data sets. It was easily evident that our system was less susceptible to topic drift and clique attacks than HITS. We do not have space to report anecdotes, so we instead focus on some quantitative measurements.

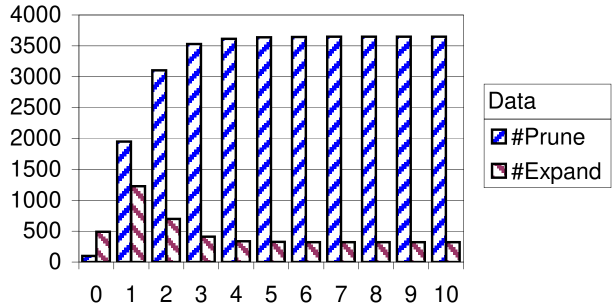


Figure 11: Our microhub smoothing technique is highly adaptive: the number of nodes pruned vs. expanded changes dramatically across iterations, but stabilizes in the end.

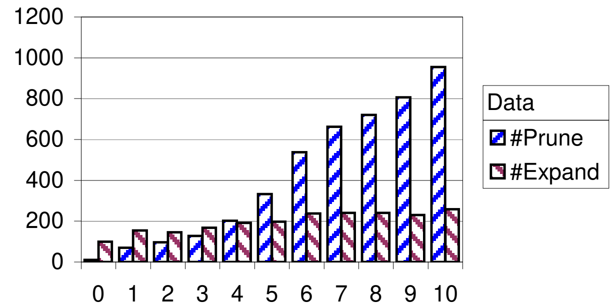


Figure 12: For some queries for which HITS showed high drift, our algorithm continues to expand a relatively larger number of nodes in an attempt to suppress drift.

DMoz and classification: First we trained the RAINBOW classifier on all the topics of DMoz. Next we picked some topics and used the example URLs in DMoz to seed the graph to be analyzed. As in HITS, we also included pages distant from these seed pages by one hyperlink. This graph is subjected to topic distillation. The 40 top authorities from the distillation algorithm (HITS, DOMHITS or DOMTEXTHITS) are submitted to the classifier. For each authority document d (which may not belong to the examples compiled by DMoz), the classifier returns a Bayesian estimate of $\Pr(c|d)$. These are added up as the *expected number of relevant authorities* among the top 40. Figure 10 clearly shows that the topical “purity” is best upheld by DOMTEXTHITS, followed by DOMHITS, HITS being the worst in this regard.

Pruning vs. expansion: In Figures 11 and 12 we plot relative numbers of nodes pruned vs. expanded against the number of iterations. Queries which do not have a tendency to drift look like Figure 11. Initially, both numbers are small. As the system bootstraps into controlled authority diffusion, more candidate hubs are pruned, i.e., accepted in their entirety. Diffused authority scores in turn lead to fewer nodes getting expanded. For queries with a strong tendency to drift (Figure 12), the number of nodes expanded does not drop as low as in low-drift situations. It is significant that the algorithm adapts to low- and high-drift pages automatically. For all the 28 queries, the respective counts stabilize within 10–20 iterations.

5 Summary and ongoing work

We believe that the application of machine learning to DOM structure will be valuable in bridging the gap between a schema-less, chaotic view of the Web and a relatively structured data model. In this paper we have illustrated our claim with two examples. In ongoing work, we are extending the fine-grained crawler to exploit link patterns across pages and longer paths. We are also trying to speed up our fine-grained topic distillation by factoring out site-level templates common to many pages.

References

[1] K. Bharat and M. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *21st International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 104–111, Aug. 1998. Online at `http://www.henzinger.com/monika/mpapers/sigir98_ps.ps`.

[2] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the 7th World-Wide Web Conference (WWW7)*, 1998. Online at `http://decweb.ethz.ch/WWW7/1921/com1921.htm`.

[3] S. Chakrabarti. Integrating the document object model with hyperlinks for enhanced topic distillation and information extraction. In *WWW 10*, Hong Kong, May 2001. Online at http://www10.org/cdrom/papers/489.

[4] S. Chakrabarti, B. E. Dom, S. Ravi Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, D. Gibson, and J. Kleinberg. Mining the Web's link structure. *IEEE Computer*, 32(8):60–67, Aug. 1999.

[5] S. Chakrabarti, M. M. Joshi, and V. B. Tawde. Enhanced topic distillation using text, markup tags, and hyperlinks. In *SIGIR*, volume 24, New Orleans, USA, Sept. 2001. ACM.

[6] S. Chakrabarti, K. Punera, and M. Subramanyam. Accelerated focused crawling through online relevance feedback. In *WWW*, Hawaii, may 2002. ACM.

[7] S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: a new approach to topic-specific web resource discovery. *Computer Networks*, 31:1623–1640, 1999. First appeared in the 8th International World Wide Web Conference, Toronto, May 1999. Available online at http://www8.org/w8-papers/5a-search-query/crawling/index.html.

[8] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley and Sons, Inc., 1991.

[9] M. Diligenti, F. Coetzee, S. Lawrence, C. L. Giles, and M. Gori. Focused crawling using context graphs. In A. E. Abbadi, M. L. Brodie, S. Chakravarthy, U. Dayal, N. Kamel, G. Schlageter, and K.-Y. Whang, editors, *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, pages 527–534. Morgan Kaufmann, 2000. Online at http://www.neci.nec.com/~lawrence/papers/focus-vldb00/focus-vldb00.pdf.

[10] B. Kahle. Preserving the internet. *Scientific American*, 276(3):82–83, Mar. 1997. Online at http://www.sciam.com/0397issue/0397kahle.html and http://www.alexa.com/~brewster/essays/sciam_article.html.

[11] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *JACM*, 46(5):604–632, 1999.

[12] S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Semi-automated Web taxonomy construction. Research Report, 2002. Online at http://www.verity.com/pdf/taxonomy.pdf.

[13] S. Lawrence and C. Lee Giles. Accessibility of information on the Web. *Nature*, 400:107–109, July 1999.

[14] A. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. Software available from http://www.cs.cmu.edu/~mccallum/bow/, 1998.

[15] J. Rennie and A. McCallum. Using reinforcement learning to spider the web efficiently. In *ICML*, 1999. Online at http://www.cs.cmu.edu/~mccallum/papers/rlspider-icml99s.ps.gz.

[16] J. Rissanen. Stochastic complexity in statistical inquiry. In *World Scientific Series in Computer Science*, volume 15. World Scientific, Singapore, 1989.

[17] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[18] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 1979. Online at http://www.dcs.gla.ac.uk/Keith/Preface.html.