Template Detection via Data Mining and its Applications

Ziv Bar-Yossef*
Computer Science Division
University of California at Berkeley
387 Soda Hall
Berkeley, CA 94720-1776, USA
zivi@cs.berkeley.edu

Sridhar Rajagopalan
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120, USA
sridhar@almaden.ibm.com

ABSTRACT

We formulate and propose the template detection problem, and suggest a practical solution for it based on counting frequent item sets. We show that the use of templates is pervasive on the web. We describe three principles, which characterize the assumptions made by hypertext information retrieval (IR) and data mining (DM) systems, and show that templates are a major source of violation of these principles. As a consequence, basic "pure" implementations of simple search algorithms coupled with template detection and elimination show surprising increases in precision at all levels of recall.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval

General Terms

Algorithms

Keywords

Information Retrieval, Hypertext, Web Searching, Data Mining

1. INTRODUCTION

This paper deals with a novel application of the classical (see Agrawal and Srikant [1]) *frequent item set* based data mining paradigm to the area of search and mining of web data. We also address an interesting software architecture issue, namely, what should be the boundary between crawling (or data gathering) systems and ranking, mining or indexing (generically, data analysis) systems. We take the point of view that the latter data analysis tools should be both clean in concept and simple in implementation, and should not, in particular, include logic for dealing with many special cases. It should be the responsibility of the data gathering system to provide clean data. This boundary is important—the effort involved in cleaning the raw data can be effectively leveraged by a large number of data analysis tools.

*Part of this work was done while the author was visiting IBM Almaden Research Center. Supported by NSF Grant CCR-9820897.

WWW2002, May 7–11, 2002, Honolulu, Hawaii, USA.
ACM 1-58113-449-5/02/0005.

In the following pages, we describe a set of three well understood principles, the Hypertext IR Principles, which may be thought of as the basic tenets of a *contract* between gathering subsystems and analytic ones. We justify these principles based on three case studies drawn from the web search and mining literature. We then apply these principles to define the *template detection* problem and show that it is an instance of the frequent item set counting problem which is well studied in the classical data mining world. However, the size and scale of the problem preclude the use of existing algorithmic methods (such as *a priori* or the *elimination generation method*) and necessitates the invention of new ones. We discuss one solution which is very effective in practice. We show that applying our method results in significant improvements in precision across a wide range of recall values. We feel that similar improvements should follow for most if not all other hypertext based search and mining algorithms.

The context:. What is the appropriate interface between data gathering and data processing in the context of web search and mining systems? (a) Modern hidden web and focused crawlers, as well as (b) Requirements imposed by the `robots.txt` robots exclusion and politeness protocols have meant the incorporation of sophisticated queuing and URL discovery methods into the crawler. Increasingly, data cleaning tasks such as mirror and duplicate detection have been transferred to gathering subsystems.

The new shift marks a change in the software engineering of web search and mining systems wherein increasingly "global" analysis is being expected of gathering subsystems. On one hand, this increases the complexity of crawling systems. On the other, it raises the possibility of multiple data mining algorithms reaping the benefits of a common data cleaning step. We will discuss this issue further in Section 1.1.

Templates and data mining:. In this paper, we will focus on a particularly interesting and recently emerging issue which impacts a number of algorithmic methods. Many websites, especially those which are professionally designed, consist of templatized pages. A templatized page is one among a number of pages sharing a common administrative authority and a look and feel. The shared look and feel is very valuable from a user's point of view since it provides context for browsing. However, templatized pages skew ranking, IR and DM algorithms and consequently, reduce precision.

In almost all instances, pages sharing a template also share a large number of common links. This is not surprising since one of the main functions of a template is in aiding navigation. If we consider web pages as items, and the set of web pages cited on any particular web page as an item set, then a frequent item set corresponds to a template. Thus, finding templates can be viewed as an instance of the frequent itemset counting paradigm. The size of the web, however, precludes using *a priori* [1] to identify template instances. Alternative methods such as the *elimination generation method* [20] also fail.

1.1 Context

The basic principles:. Despite their differences in detail, there are three important principles (or assumptions)—which we call the *Hypertext IR Principles* henceforth—underlying most, if not all, reference based (or hypertext) methods in information retrieval. We do not intend that these are exclusive, but that these cover almost all the algorithmic uses that links have been put to.

1. **Relevant Linkage Principle**: Links point to relevant resources. The Relevant Linkage Principle is recognized implicitly in early work such as Garfield's *impact factor* [16], and in subsequent work in bibliometrics. The tagline in Garfield's paper states: *Journals can be ranked by frequency and impact of citations for science policy studies.* In particular, Pinski and Narin's *influence factor* [24] weights citations by the influence value of the citing page. An instance of this principle is that links *confer authority* [19]: To quote [19],

   > The creation of a link on the WWW represents the following type of judgment: the creator of page p by linking to page q has to some measure conferred authority on q.

   Brin and Page [3, 23] in defining the PageRank ranking metric say that links express a similar motive:

   > The intuition behind PageRank is that it uses information external to the pages themselves – their backlinks, which provides a kind of peer review.

2. **Topical Unity Principle**: Documents often co-cited are related, as are those with extensive bibliographic overlap. Small, in the context of academic literature [26], observed that documents which are cited together are relevant to each other. Thus, one can use co-citation as a way of spreading activation [25] from one document to another. The flip side of this observation is an earlier observation of Kessler [18]. Kessler uses bibliographic overlap as a measure of mutual relevance. While bibliographic overlap is easier to measure than co-citation strength, it is more brittle and given to manipulation by "spamming," a practice which is becoming increasingly common on the web. Modern IR tools, such as HITS, Clever [10, 8], and SALSA [21] make extensive use of both co-citation strength and bibliographic overlap as a measure of mutual relevance between documents.

3. **Lexical Affinity Principle**: Proximity of text and links within a page is a measure of the relevance of one to the other [22]. Vannevar Bush anticipated this [5]. His vision, the *memex*, resembled the human mind—making associations between the content of one document, and the location of another. This process of tying two things together involved the use of semantic keys chosen by the user, and was not simply an anonymous, uninterpretable relationship:

   > It affords an immediate step, however, to associative indexing, the basic idea of which is a provision whereby any item may be caused at will to select immediately and automatically another. This is the essential feature of the memex. The process of tying two items together is the important thing.

   An extension of the Lexical Affinity Principle is detailed in Chakrabarti's work [6]. In this work, he proposes that distance between entities within a document should be measured by considering document structure and not simply some linearization of it. For instance, the distances between all pairs of elements within an itemized list should be uniform.

Our contributions:. In this paper, we take the point of view that the Hypertext IR Principles form a sound basis for defining the contract between modern data gathering and data analysis systems. Thus, the function of a good data gathering system should be to preprocess the document corpus so that violations of the Hypertext IR Principles are minimized. This has two benefits. First, it results in better software design, with a potentially large number of different analysis and mining algorithms leveraging the effort expended in the pre-processing phase. Second, the data analysis algorithms can be retained in "pure form" and would not need to expend a lot of effort dealing with special cases.

We illustrate this principle by considering the problem caused by templatized pages. Our discussion in this context is three fold. First, we define templates and argue that templates are a source of wholesale violation of the Hypertext IR Principles. We then formulate and describe data mining algorithms to detect and flag templates. We then show that applying a "pure form" ranking algorithm (Clever, with no bells and whistles) to the template eliminated database results in remarkable improvements in precision at all recall levels across the board. We finally speculate that the same approach will work in a more general context.

1.2 Noisy data and associated problems

The web contains frequent violations of the Hypertext IR Principles. These violations are not simply random. They happen in a systematic fashion for many reasons, some of which we describe below.

Relevant Linkage Principle The web contains many *navigational links* (links that help navigating inside a website), *download links* (links to download pages for instance, those which point to the Netscape download page), links which point to business partners, links which are introduced to deliberately mislead link based searching algorithms, and paid *advertisement links*. Each such auxiliary link violates the Relevant Linkage Principle.

Topical Unity Principle A number of pages speak to a mixture of topics. A particularly frequent case in point is a bookmark page or a personal homepage. Resource pages (for instance, "links for Electrical Engineers") also tend to be topically diverse. In the context of search, whether a page is topically unified or diverse depends on the granularity of the query. For instance, the "links for Electrical Engineers" page can be topically unified if the query were "Electrical Engineering", but diverse if it were "Frequency Division Multiplexing".

Lexical Affinity Principle HTML is a linearization of a document. The true structure of it, however, is most like a tree. For constructs such as a (two dimensional) table, trees are not effective descriptions of document structure either. Thus, lexical affinity should be judged based on the real structure of the document, not on the particular linearization of it as determined by the conventions used in HTML. As an instance, lists that are arranged in alphabetical order within a page abound on the web. It would be wrong to apply lexical affinity to such a linear list.

Systematic violations of the Hypertext IR Principles result in a host of well known problems. Most search and mining engines have a lot of special purpose code to deal with each such problem. We describe some prominent classes of these problems here.

Generalization Generalization occurs when a very general resource page, such as the Yahoo homepage page is ranked as a highly authoritative page regardless of the query.

Topic Drift Topic drift is the event wherein an authoritative page within a better connected and larger, but peripheral, community of web pages gets highly ranked. A colleague is a fan of the San Francisco 49ers football team, as well as an authority on finite model theory. These two interests are obvious from his homepage. Since the web has a significantly larger amount of information about the 49ers than it has about finite model theory, it is possible, even likely, that link based search for resources about finite model theory returns pages about the San Francisco 49ers.

Bias A page lexically cited close to an authoritative page on a subject often gets artificially high scores even though the proximity is accidental. For instance, on a search for "Computing companies," "Micrografix" could be highly ranked because of its proximity to "Microsoft" in alphabetic listings.

1.3 Pagelets

Modern web pages (see, for instance, Figure 1), contain many elements for navigational and other auxiliary purposes. For example:

1. Popular web sites (e.g., search engine sites) tend to contain, in addition to the main content, a lot of auxiliary information, such as advertisement banners, shopping links, navigational bars, privacy policy information, and even news headlines.

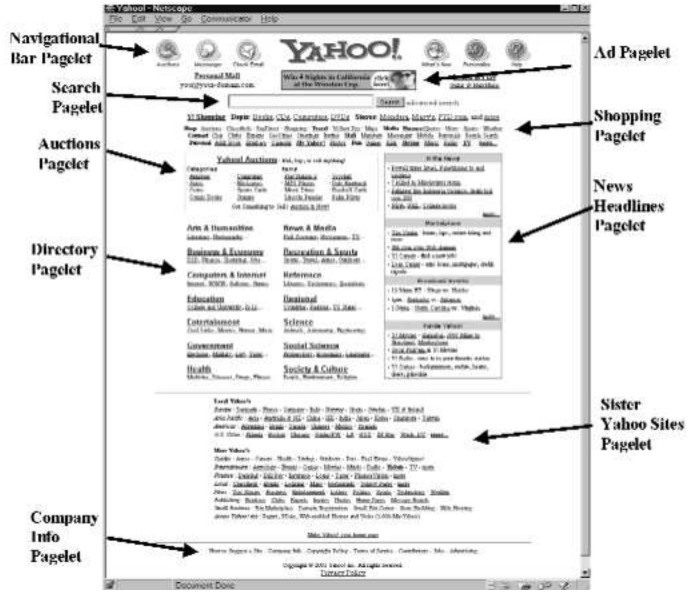

Figure 1: The Yahoo! pagelets.

2. Pages many times represent a collection of interests and ideas that are loosely knit together to form a single entity. For example, personal homepages may contain information relevant to the person's work as well as information relevant to her hobbies. Resource lists (e.g., bookmarks) sometimes include information about many topics, which are not necessarily related to each other.

This implies that much of the noise problem follows from violations of the Relevant Linkage Principle and the Topical Unity Principle that are caused by the construction of modern web pages.

A pagelet (cf., [6, 11]) is a self-contained logical region within a page that has a well defined topic or functionality. A page can be decomposed into one or more pagelets, corresponding to the different topics and functionalities that appear in the page. For example, the Yahoo! homepage, `www.yahoo.com` (see Figure 1), can be partitioned into the main directory pagelet at the center of the page, the search window pagelet above it, the navigational bar pagelet at the top, the news headlines pagelet on the side, and so forth.

We propose that pagelets, as opposed to pages, are the more appropriate unit for information retrieval. The main reason is that they are more structurally cohesive, and better aligned with both the Topical Unity Principle and the Relevant Linkage Principle. Several issues concerning the process of discovery and use of pagelets are discussed in Section 3.1.

1.4 Templates

A frequent and systematic violation of the Hypertext IR Principles is due to the proliferation in the use of *templates*. A template is a pre-prepared master HTML shell page that is used as a basis for composing new web pages. The content of the new pages is plugged into the template shell, resulting

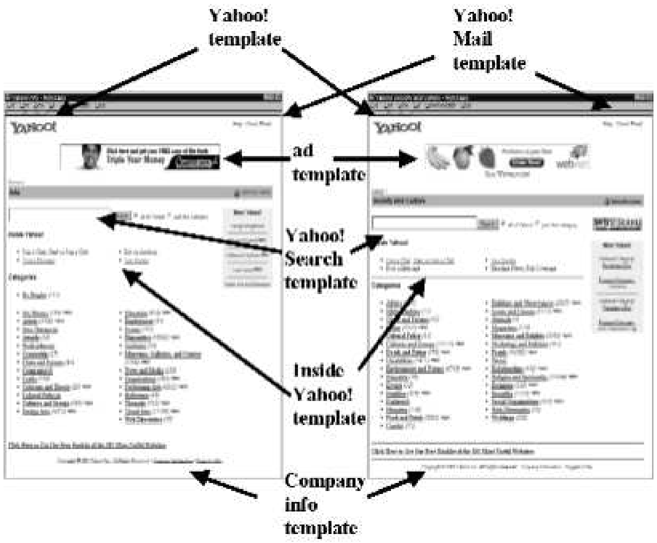

Figure 2: The Yahoo! template. As is evident from the figure, the unique content in each page is a relatively small fraction of the entire page.

in a collection of pages that share a common look and feel.

Templates can appear in primitive form, such as the default HTML code generated by HTML editors like Netscape Composer or FrontPage Express, or can be more elaborate in the case of large web sites. These templates sometimes contain extensive navigational bars that link to the central pages of the web site, advertisement banners, links to the FAQ and Help pages, and links to the web site's administrator page. See Figure 2 for a representative example: the Yahoo! template. The use of templates has grown with the recent developments in web site engineering. Many large web sites today are maintained automatically by programs, which generate pages using a small set of formats, based on fixed templates. Templates can spread over several sister web sites (e.g., a template common to `amazon.com` and `drugstore.com`), and contain links to other web sites, such as endorsement links to business partner web sites (`www.eunet.net`), advertisement links, and "download" links. Thus, traditional techniques to combat templates, like intra-site link filtering, are not effective for dealing with their new sophisticated form.

Since all pages that conform to a common template share many links, it is clear that these links cannot be relevant to the specific content on these pages. Thus templates violate both the Relevant Linkage Principle and the Topical Unity Principle. They may (and do) also cause violations of the Lexical Affinity Principle, if they are interleaved with the actual content of the pages. Therefore, improving hypertext data quality by recognizing and dealing with templates seems essential to the success of the hypertext IR tools.

A template has two characterizing properties:

(1) There is a significant collection of pages that conform to this template.

(2) This common look-and-feel of these pages is controlled or influenced by a (single) central authority.

The latter property is important in order to distinguish between templates, in which a collection of pages intentionally share common parts, and the following:

(1) mirrors—complete wholesale duplications of pages or sites and

(2) independent pages that accidentally share similar parts, which might be important signatures of communities (see [20]).

Note that the requirement that a central authority influences all pages that conform to a template does not necessarily imply that they all belong to a single web site. The central authority can also be one that coordinates templates between sister sites.

In Section 3.2 we give a formal definition of templates, and show two efficient algorithms to detect templates in a given dataset of pages. Having the ability to detect templates, we can discard them from the dataset, and thus improve its quality significantly. Our thesis is that pagelet-based information retrieval has to go hand in hand with template removal, in order to achieve the desired improvement in performance. In the applications we consider in Section 4 we pursue this direction, and show in Section 5 that this results in better performance for pagelet-based Clever.

2. THREE CASE STUDIES

In this section, we pick three modern canonical algorithms using link based analysis and, in each case, describe their reliance on the Hypertext IR Principles described earlier.

2.1 Case 1: HITS and friends

HITS [19] is a ranking algorithm for hypertext corpora. The goal is to choose, given a query t, from a base set of pages, $G(t)$, all of which are assumed to be relevant to t, the most relevant pages. This is done as follows. Start from an initial assignment $h(p) = a(p) = 1$ for each page p in $G(t)$ and iteratively perform the following update steps:

$$h(p) = \sum_{q \in O(p)} a(q) \qquad\qquad a(p) = \sum_{q \in I(p)} h(q)$$

Here, $I(p)$ and $O(p)$ denote the pages which point to and are pointed to respectively by p. When normalized after each update, $h(p)$ and $a(p)$ converge to a fixed point, known as the hub and authority score respectively. The higher these scores, the more relevant the pages to the query term t.

1. HITS implicitly depends the Relevant Linkage Principle in that it does not distinguish between links on the page. Kleinberg did note the bad effects caused by local (or nepotistic) links and excluded them altogether from $I(p)$ and $O(p)$.

2. HITS implicitly assumes that pages satisfy the Topical Unity Principle. Indeed, Kleinberg was aware of this as well, and pointed out that in cases where pages were topically diverse, both *generalization* and *topic drift* can and do occur. The second effect, known as the TKC effect, was more completely described and addressed in the work of [21] by dynamically partitioning pages in $G(t)$ into topically cohesive units.

3. HITS ignored the Lexical Affinity Principle both in its application to the text surrounding links and in its application to affinity between links occurring close to each other within a page. This issue was addressed in [2, 7] where in each case links were weighted by the relevance of the text surrounding it to the query t. In

addition, weight was propagated more aggressively between proximal links than through distant ones. While generally beneficial, results obtained using these methods were unsatisfactory when pages in $G(t)$ violated the Lexical Affinity Principle.

Discussion. The methods (2) suggested in [21] and (3) suggested in [2, 7] require significant query time processing, which *per se*, would not be required if the data were preprocessed to satisfy the Topical Unity Principle and the Lexical Affinity Principle respectively. Moreover, if a more sophisticated method, such as that suggested by Davison [14] were used to clean the data beforehand, the (somewhat unsatisfactory) hack (1) used in [19] would not be required either.

2.2 Case 2: Focused crawling

A focused crawler is a program that looks for pages that are relevant to some node in a given taxonomy. The notion of focused crawling was introduced and first discussed in [12, 13]. The solution proposed (called FOCUS) consisted of three components: a crawler, a classifier, and a distiller. The three components work in concert as follows: The function of the crawler was to fetch pages from the web according to a given priority. The classifier locates these fetched pages within the taxonomy, and the distiller tags pages in each taxonomy node with a relevance score, which is then used to prioritize those unfetched pages that are pointed to by highly relevant nodes for fetching by the crawler.

The distiller used by FOCUS was Clever, which is a variant of Kleinberg's HITS Algorithm. The classifier used was HyperClass, described in [9].

1. Since FOCUS uses Clever, the analysis in Section 2.1 applies to FOCUS as well.

2. The crawler assumes that all pages pointed to by highly relevant pages are good pages to fetch, which is essentially the same as assuming that the Relevant Linkage Principle and the Topical Unity Principle hold for these pages.

3. HyperClass uses a Markov Random Field approach to classify pages into a given taxonomy. HyperClass (as described in [9]) assumes all the Hypertext IR Principles.

Discussion. The measure of effectiveness of a focused crawler is (1) its harvest rate—the fraction of pages that are downloaded by it which are relevant to some node in the taxonomy and (2) the accuracy with which these pages are located within the taxonomy. Following (1), a good distiller would result in better prioritization of the pages to be fetched, and thus directly improve harvest rate. Moreover, if the data would adhere more closely to the Relevant Linkage Principle and the Topical Unity Principle following (2), pointers to irrelevant material would not be followed, and would directly improve harvest rate. Finally, any improvements to HyperClass (3) accruing from cleaner data would automatically improve the accuracy of the focused crawler.

2.3 Case 3: The co-citation algorithm

The co-citation algorithm is a simple algorithm due to Dean and Henzinger [15]. Given a page p, the algorithm finds all pages which are "similar" in content by looking for other pages that are most often co-cited with p.

At its heart the co-citation algorithm is an immediate consequence of assuming the Topical Unity Principle and the Relevant Linkage Principle.

Discussion. Violations of the Topical Unity Principle and the Relevant Linkage Principle, for instance, because a popular site pointed to for irrelevant reasons, skew the results of the co-citation algorithm.

3. ALGORITHMS FOR PAGE PARTITIONING AND TEMPLATE DETECTION

In this section we describe our algorithms for partitioning a given web page into pagelets and for detecting templates in a given collection of pages.

3.1 Page Partitioning Algorithm

Before we describe our algorithm for partitioning a web page into pagelets, we would like to define what pagelets are. The following is a *semantic* definition of pagelets:

DEFINITION 1 (PAGELET - SEMANTIC DEFINITION). *A* pagelet *is a region of a web page that (1) has a single well-defined topic or functionality; and (2) is not nested within another region that has exactly the same topic or functionality.*

That is, we have two contradicting requirements from a pagelet: (1) that it will be "small" enough as to have just one topic or functionality; and (2) that it will be "big" enough such that no other region that contains it also has the same topic (the nesting region may have a more general topic). Figure 1 demonstrates Definition 1 by the pagelet partitioning of the Yahoo! homepage.

In order to partition a page into pagelets, we need a *syntactic* definition of pagelets, which will materialize the intuitive requirements of the semantic definition into an actual algorithm. This problem was considered before by Chakrabarti [6] and Chakrabarti *et al.* [11]; they suggested a sophisticated algorithm to partition "hubs" in the context of the HITS/Clever algorithm into pagelets. Their algorithm, however, is intertwined with the application (Clever) and in particular it is *context-dependent*: the partitioning depends on the given query. Furthermore, the algorithm itself is nontrivial, and therefore it seems infeasible to run it on millions of pages at the data gathering phase.

Since our principal goal is to design efficient hypertext cleaning algorithms that run in data gathering time, we adopt a simple heuristic to syntactically define pagelets. This definition has the advantages of being context-free, admitting an efficient implementation, and approximating the semantic definition quite faithfully. Our heuristic uses the cues provided by HTML mark-up tags such as tables, paragraphs, headings, lists, etc.

The simplest approach to use HTML, and more generally XML, in page partitioning is to define the HTML elements (the "tags") as the pagelets. However, this approach suffers from several caveats: (1) the HTML structure is a tree and we would like a flattened partitioning of the page; and (2) the granularity of this partitioning is too fine, and does not meet the second requirement of Definition 1. We refine the approach in the following way:

```
Page Partitioning Algorithm

Partition(p) {

T_p <--- HTML parse tree of p

Queue <--- root of T_p

while (Queue is not empty) {

      v <--- top element in Queue

      if (v has a child with at least k links)
            push all the children of v to Queue
      else
            declare v as a pagelet
}
```

Figure 3: Page partitioning algorithm.

Definition 2 (Pagelet - syntactic definition). *An HTML element in the parse tree of a page p is a pagelet if (1) none of its children contains at least k hyperlinks; and (2) none of its ancestor elements is a pagelet.*

The first requirement in Definition 2 corresponds to the first requirement of Definition 1. When an HTML element contains at least k links (k is a parameter of our choice; in our implementation we use $k = 3$), then it is likely to represent some independent idea/topic; otherwise, it is more likely to be topically integrated in its parent. The second requirement in Definition 2 achieves two goals: first, it implies a flattened partitioning of the page, and second, it ensures the second requirement of Definition 1. Definition 2 implies the page partitioning algorithm depicted in Figure 3.

3.2 Template Detection Algorithms

We now present two algorithms for detecting templates in a given collection of pages. Both algorithms are scalable and designed to process large amounts of pages efficiently. The first algorithm, called the *local template detection algorithm*, is more accurate for small sets of pages, while the second algorithm, called the *global template detection algorithm*, better suits large sets of pages.

Before we describe the algorithms, we define templates formally.

Definition 3 (Template - semantic definition). *A* template *is a collection of pages that (1) share the same look and feel and (2) are controlled by a single authority.*

Templates are usually created by a master HTML page that is shared by all the pages that belong to the site(s) of the authority that controls the template. The specific content of each page is inserted into "place holders" in this master HTML page. The outcome of this process is that all the templatized pages share common pagelets, such as navigational bars, ad banners, and logo images. The Yahoo! template, depicted in Figure 2, provides a good example of this notion.

The second requirement in Definition 3 is crucial in order to distinguish between templates and (a) whole-sale duplications of pages and (b) accidental similarities between independent pages.

In order to materialize the semantic definition of templates, we use the following syntactic definition. In the definition, we use the notation $O(p)$ to denote the page owning a pagelet p, and $C(p)$ to denote the content (HTML content) of the pagelet p.

Definition 4 (Template - syntactic definition). *A* template *is a collection of pagelets $p_1, \ldots, p_k$ that satisfies the following two requirements:*
(1) $C(p_i) = C(p_j)$ for all $1 \leq i \neq j \leq k$.
(2) $O(p_1), \ldots, O(p_k)$ form an undirected connected component.

Definition 4 associates a template with the collection of pagelets that are shared by the templatized pages. The first requirement in Definition 4 ensures that the template pagelets are indeed part of the common "look and feel" of the templatized pages. The second requirement in Definition 4 ensures the second requirement of Definition 3. We use here the heuristic that templatized pages that are controlled by a single authority are usually reachable one from the other by an undirected path of other templatized pages (possibly through the root of the site). On the other hand, mirror sites and pages that share accidental similarities are not likely to link to each other.

In practice, we relax the first requirement of Definition 4, and require that pagelets in the same template are only "approximately" identical. That is, if every two pagelets in a collection $p_1, \ldots, p_k$ are almost-identical and their owner pages form a connected component, we consider them to be a template. The reason for this relaxation is that in reality many templatized pages are slight perturbations of each other, due, e.g., to version inconsistencies. We use the shingling technique of Broder *et al.* [4] to determine almost-similarities. A *shingle* is a text fingerprint that is invariant under small perturbations. We associate with each template $\mathcal{T}$ a *shingle* $s(\mathcal{T})$, and require each pagelet $p \in \mathcal{T}$ to satisfy $s(p) = s(\mathcal{T})$.

We further denote by $O(\mathcal{T})$ the collection of pages owning the pagelets in $\mathcal{T}$.

The algorithmic question our template detection algorithms are required to solve is as follows: given a collection of hyperlinked documents $G = \langle V_G, E_G \rangle$ drawn from a universe $W = \langle V_W, E_W \rangle$ (i.e., G is a sub-graph of W), enumerate all the templates $\mathcal{T}$ in W for which $O(\mathcal{T})$ intersects V_G. The algorithms assume the pages, links, and pagelets in G are stored as local database tables. We will show that their running time and space requirements are small—at most quasi-linear in the size of G.

The algorithms we present assume that G is stored as the following database relations:

- `PAGES(page_key,page_shingle)` - the pages in G together with their shingle values.
- `LINKS(src_page_key,dest_page_key)` - the hyperlinks between the pages in the dataset.
- `PAGELETS(page_key,pagelet_serial,pagelet_shingle)` - the list of pagelets in each page together with their shingle values.

Building these tables can be done by streaming through the pages in the dataset, and parsing their HTML text to

Local Template Detection Algorithm

(1) Eliminate duplicate pages in G

(2) Sort and group the pagelets in G according to their shingle. Each such group represents a template.

(3) Enumerate the groups, and output the pagelets belonging to each group.

Figure 4: Local template detection algorithm.

extract their links and pagelets. Thus, it requires a constant time per page.

The first algorithm we consider is more suitable for document sets that consist only of a small fraction of the documents from the larger universe. In such sets it is very probable for a template $\mathcal{T}$ that only a small fraction of the pages in $O(\mathcal{T})$ are contained in V_G. This implies that, although $O(\mathcal{T})$ forms an undirected connected component in W, $O(\mathcal{T}) \cap V_G$ may not be connected in G. In this case, therefore, it makes more sense to validate only the first requirement of Definition 4. Note that for small sets of pages, we do not have to worry about accidental page similarities, because they are unlikely to occur. Whole-sale duplication of pages can be detected at a pre-processing step using the Broder *et al.* algorithm. The algorithm is described in Figure 4.

To analyze the algorithm's complexity, we denote by N the number of pages in G, by M the number of links, and by K the number of pagelets in G. N, M, and K are the sizes of the tables `PAGES, LINKS`, and `PAGELETS` respectively.

The algorithm is very efficient: the first step can be carried out in $O(N \log N)$ time by computing the shingles of each pages, and sorting the pages according to the shingle value. Similarly, the second step requires $O(K \log K)$ steps, and the third step is just an $O(K)$ linear scan of the pagelets. The space requirements (on top of the database tables) are logarithmic in K and N. Note that the operations performed in this algorithm (sorting and enumerating) are very standard in databases, and therefore are highly optimized in practice.

Our second algorithm, depicted in Figure 5, is more involved, and detects only templates that satisfy both requirements of Definition 4. This algorithm is well suited for large subsets of the universe, since in such subsets $O(\mathcal{T}) \cap V_G$ is likely to be connected in G.

The algorithm works in two phases: in the first phase it extracts all the groups of syntactically similar pagelets (steps (1)-(2)) and in the second phase it partitions each such group into templates (steps (3)-(5)).

The main point here is that if $\mathcal{T}$ is a template that intersects V_G, then $s(\mathcal{T})$ will be one of the shingles in the relation `TEMPLATE_SHINGLES`. The group of the shingle $s(\mathcal{T})$ should contain all the pagelets in $\mathcal{T}$, but may contain pagelets from other pages, whose shingle happens to coincide with $s(\mathcal{T})$. We extract the pagelets of $\mathcal{T}$ from the shingle's group using the connected components algorithm: the owners of the pagelets in $\mathcal{T}$ should form an undirected connected component.

This algorithm is also efficient: step (1) requires sorting `PAGELETS`, which takes $O(K \log K)$ steps and $O(\log K)$ space. Step (2) is just a linear scan of `PAGELETS`. Note that after this step the size of `PAGELETS` may decrease significantly, because we eliminate all the pagelets that occur only once in the dataset.

In step (3) we perform a triple join `TEMPLATE_CANDIDATES T1` $\bowtie$ `TEMPLATE_CANDIDATES T2` $\bowtie$ `LINKS` with constraints

1. `T1.page_key = LINKS.src_page_key`

2. `T2.page_key = LINKS.dest_page_key`

3. `T1.pagelet_shingle = T2.pagelet_shingle`

Computing the join takes $O(d^2 M \log(d^2 M))$ time and space, where d is the maximal number of template pagelets in a page. Note that d is a fixed small number. The size of the outcome of step (3), `TEMPLATE_LINKS`, is at most $O(d^2 M)$.

In step (4) we assume that for every s, G_s is small enough to store in main memory. This is a reasonable assumption, since usually the size of a template is no more than a few thousands of pages. Thus even if G_s consists of several templates, we can store the few (tens) of thousands of records in main memory. This allows us to run a BFS algorithm on G_s, which would have been prohibitive if G_s was stored on disk. Thus, steps (4)-(5) take at most $O(d^2 M)$ steps and logarithmic space.

All in all, the algorithm takes $O(K \log K + d^2 M)$ steps and $O(d^2 M)$ space.

4. THREE CASE STUDIES REVISITED

We now revisit the three case studies from the point of view of incorporating knowledge of pagelets and templates into them. We do this primarily to establish that the modifications required of the original algorithms are simple, minimal and natural.

Case 1: HITS and Clever. Instead of using a graph consisting of all links (or all non-nepotistic links), we construct a graph over two sets of vertices. There is a vertex corresponding to each non-template pagelet in the base set, and another corresponding to each page in the base set. Edges are directed from vertices corresponding to pagelets to vertices corresponding to pages in the natural way, i.e., if and only if the pagelet contains a link to the page. Edges out of template pagelets are omitted entirely. The HITS/Clever algorithms can now be run as is. Hubs will always be pagelets, and authorities pages.

Case 2: Focused Crawling. The focused crawler can use template and pagelet information in two ways.

(1) The distiller is modified exactly as described in the case of the HITS algorithm above.

(2) The second change concerns the crawler. Pages which are pointed to out of template pagelets can be assigned a reduced priority for crawling.

Case 3: The co-citation algorithm. The co-citation algorithm can exploit pagelet and template information in two ways.

(1) Co-citation is counted only if it occurs within the same pagelet, not simply on the same page. Alternately,

Global Template Detection Algorithm

(1) Select all the pagelet shingles in `PAGELETS` that have at least two occurrences. Call the resulting table `TEMPLATE_SHINGLES`. These are the shingles of the re-occurring pagelets.

(2) Extract from `PAGELETS` only the pagelets whose shingle occurs in `TEMPLATE_SHINGLES`. Call the resulting table `TEMPLATE_CANDIDATES`. These are all the pagelets that have multiple occurrences in G.

(3) For every shingle s that occurs in `TEMPLATE_SHINGLES` define G_s to be the shingle's group: all the pages that contain pagelets whose shingle is s. By joining `TEMPLATE_CANDIDATES` and `LINKS` find for every s all the links between pages in G_s. Call the resulting relation `TEMPLATE_LINKS`.

(4) Enumerate the shingles s in `TEMPLATE_SHINGLES`. For each one, load into main memory all the links between pages in G_s.

(5) Use a BFS algorithm to find all the undirected connected components in G_s. Each such component is either a template or a singleton. Output the component if it is not a singleton.

Figure 5: Global template detection algorithm.

co-citation within a pagelet can be weighted more heavily than that within the same page.

(2) Citations from template pagelets are entirely ignored.

5. EXPERIMENTAL RESULTS

In this section we demonstrate the benefit of template detection in improving precision of search engine results, by presenting experimental results of running the template / pagelet based implementation of Clever from Section 4.

In the experiments we compare six versions of Clever. Two of the versions use the classical "page-based" Clever, and the other four use the "pagelet-based" Clever of Section 4. The versions differ in the cleaning steps performed on the base set of pages/pagelets before starting to run the reinforcement hubs and authorities algorithm on this base set. We consider two cleaning steps: (1) FNM—filtering pages/pagelets that do not match the query term (i.e., the query term does not occur in their text), and (2) FT—filtering template-pagelets. FNM is applicable to both the page-based Clever and the pagelet-based Clever. FT is applicable only to pagelet-based Clever. Note the inherent difference between these two cleaning steps: FNM is query-dependent and can be performed only in query time, while FT is query-independent and can thus be carried out as a pre-processing step. The six Clever configurations are the following:

(1) *Page vanilla*
(2) *Page FNM*
(3) *Pagelet vanilla*
(4) *Pagelet FNM*
(5) *Pagelet FT*
(6) *Pagelet FT FNM*

For detecting the template pagelets, we use the local template detection algorithm of Section 3.2, since the algorithm is given a fairly constrained collection of pages (the base set of pages).

Our main measure of concern for comparing these Clever versions is *precision*—the fraction of search results returned that are indeed directly relevant to the query. Since the main aim of template elimination is to clean the hypertext data from "noise", the measure of precision seems the most appropriate for testing the effectiveness of template elimination.

In order to determine relevance of search results to the query term, we had to employ human judgment. We scanned each of the results manually, and classified them as "relevant" or "non-relevant". A page was deemed relevant if it was a reasonable *authority* about the query term; pages that were indirectly related to the query term were classified as non-relevant.

We used the extended ARC suite of queries [10] to test the six Clever versions. The suite consists of the following 37 queries: "affirmative action", "alcoholism", "amusement parks", "architecture", "bicycling", "blues", "cheese", "citrus groves", "classical guitar", "computer vision", "cruises", "Death Valley", "field hockey" "gardening", "graphic design", "Gulf war", "HIV", "Java", "Lipari", "lyme disease", "mutual funds", "National parks", "parallel architecture", "Penelope Fitzgerald", "recycling cans", "rock climbing", "San Francisco", "Shakespeare", "stamp collecting", "sushi", "table tennis", "telecommuting", "Thailand tourism", "vintage cars", "volcano", "zen buddhism", and "Zener".

In the experiments we ran each of the six Clever versions on the ARC set of queries, and recorded the top 50 authorities found. We then manually tagged each of the results as "relevant" or "non-relevant". We determined for each run the "precision@N" for N = 10,20,30,40,50; that is, the fraction of the top N results that were tagged as "relevant".

The chart in Figure 6 presents the average precision@N (N = 10,20,30,40,50) over the 37 ARC queries for each of

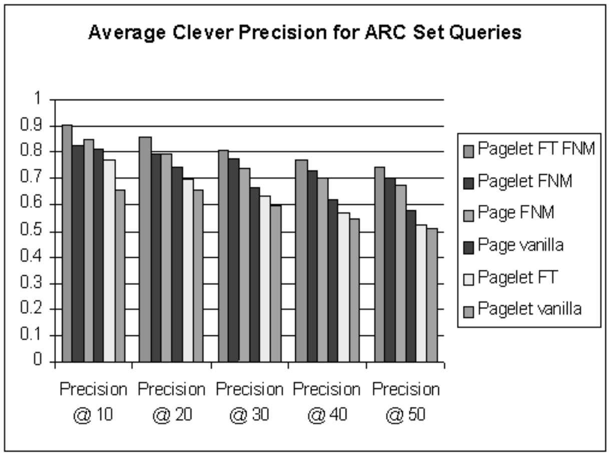

Figure 6: Average precision@N (N = 10,20,30,40,50) over the 37 ARC queries for each of the six Clever versions. The results clearly indicate that template elimination coupled with filtering non-matching pages/pagelets yields the most accurate results. For example, it improves the precision of page vanilla and page FNM at the top 10 from 81% and 85%, respectively, to 91%. The improvement at the top 50 is even more dramatic: from 58% and 67%, respectively, to 74%.

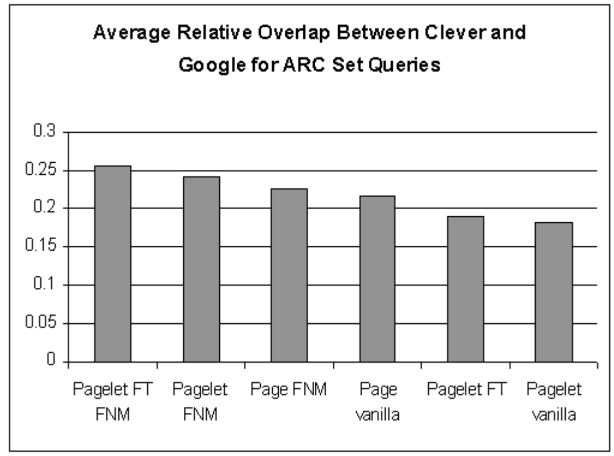

Figure 7: Average relative overlap of the results of each of the six Clever versions with the top 50 results of Google, over the 37 ARC set queries. Here, we use Google as a benchmark, and seek at maximize the overlap with its results. A large overlap indicates both high precision and high recall. The results indicate again that template elimination coupled with filtering non-matching pagelets yields the most accurate and qualitative results, but avoiding the filtering of non-matching pagelets is worse than using the page-based Clever.

the six Clever versions. The results clearly indicate that template elimination coupled with filtering non-matching pages/pagelets yields the most accurate results. For example, it improves the precision of page vanilla and page FNM at the top 10 from 81% and 85%, respectively, to 91%. The improvement at the top 50 is even more dramatic: from 58% and 67%, respectively, to 74%.

Reviewing the results manually reveals the reason for the improvements: page-based Clever, especially when running on broad topic queries for which there are many authoritative pages that reside within large commercial (and often templatized) web-sites, tends to drift towards the densely connected artificial communities created by templates (a phenomenon called in [21] the "TKC Effect"). The filtering of non-matching pages does not always circumvent this problem, because sometimes the query term itself occurs in the template, and thus all the templatized pages contain the query too, whether or not they are indeed relevant to the query.

An anecdotal example of this phenomenon is illustrated by the results of Page FNM Clever on the query "Java". Table 1 lists those of the results that were classified as "non-relevant". Note that six out of the 18 non-relevant results belonged to the "ITtoolbox" domain. The ITtoolbox domain has many child sites that discuss various information technology tools; one of them is `java.ittoolbox.com`, which was indeed returned as result no. 22. However, each of the ITtoolbox sites (including the Java one) share a template, which contains a navigational bar with links to all the sister ITtoolbox sites. The high linkage between the ITtoolbox sites caused Clever to experience a TKC Effect. Note that the filtering of non-matching pages did not filter all the sister ITtoolbox sites from the base set, because all of them contain the term "Java" (in the template navigational bar). The Pagelet FT FNM Clever, on the other hand, detected the ITtoolbox template, eliminated it from the base set, and therefore returned only `java.ittoolbox.com` as one of the results. In total, Pagelet FT FNM Clever had a precision of 74% at the top 50 for the query "Java", compared to only 64% of Page FNM Clever.

The chart of Figure 6 also indicates, however, that template elimination alone is not sufficient. In fact, the pagelet-based Clever with template elimination but without filtering of non-matching pagelets is doing worse than the classical page-based Clever. Pagelet-based Clever with no cleaning steps is doing the worst of all. These results are not surprising and they were noted before in [6]. Templates are just one source of noise in hyperlinked environments; frequently, many non-template pagelets in the base set are not relevant to the query. When such pagelets are not filtered from the base set, they magnify the ratio of noise to data significantly (because now each non-relevant page contributes many non-relevant pagelets to the base set), thereby causing the Clever algorithm to diverge. However, when most of the non-relevant pagelets are filtered from the base set, pagelet-based Clever is superior to page-based Clever, as demonstrated by the fact that the precision of the Pagelet FNM Clever (i.e., no template elimination) was better than that of Page FNM Clever.

The chart in Figure 7 shows the average relative overlap of the Clever results with the top 50 results of Google [17], over the 37 ARC set queries. Here, we use Google as a benchmark, and seek at maximizing the overlap with its results. A large overlap indicates both high precision and high recall. The results presented in this chart are consistent with the precision results presented before: template elimination coupled with filtering non-matching pagelets yields the most

#	Title	URL
27.	Sun Microsystems	www.sun.com
29.	HTML Goodies Home Page	www.htmlgoodies.com
30.	Linux Enterprise Ausgabe 11 2001 November	www.linuxenterprise.de/
31.	DevX Marketplace	marketplace.devx.com
32.	Der Entwickler Ausgabe 6 2001 November Dezember	www.derentwickler.de/
33.	**ITtoolbox Knowledge Management**	knowledgemanagement.ittoolbox.com
34.	EarthWeb com The IT Industry Portal	www.developer.com
35.	entwickler com	www.entwickler.com
36.	**ITtoolbox EAI**	eai.ittoolbox.com
38.		www.xml-magazin.de/
39.	DevX	www.devx.com
41.	The Hot Meter	www.thehotmeter.com
43.	HTML Clinic	www.htmlclinic.com
45.	**ITtoolbox Networking**	networking.ittoolbox.com
46.	FontFILE fonts...	www.fontfile.com
47.	**ITtoolbox Data Warehousing**	datawarehouse.ittoolbox.com
49.	**ITtoolbox Portal for Oracle**	oracle.ittoolbox.com
50.	**ITtoolbox Home**	www.ittoolbox.com

Table 1: Non-authoritative results output by Page FNM Clever for "Java". Note that the results include six non-relevant pages from the Information Technology site ITtoolbox. All of them belong to a template of the site.

accurate and qualitative results, but avoiding the filtering of non-matching pagelets is worse than using the page-based Clever.

Template Frequency. In order to measure how common templates are, we checked what fraction of pages in the base set of each query contained template pagelets. The results presented in Table 2 demonstrate how common and fundamental this phenomenon has become. An interesting aspect of these results is that they give an indication for each given query how pervasive and commercialized its presence on the web is.

6. CONCLUSIONS

In this paper we discussed the problem of detecting templates in large hypertext corpora, such as the web. We identified three basic principles, the Hypertext IR Principles, that underly most hypertext information retrieval and data mining systems, and argued that templates violate all three of them. We showed experimentally that templates are a pervasive phenomenon on the web, thus posing a significant obstacle to IR and DM systems. We proposed a new approach for dealing with violations of the Hypertext IR Principles, in which the hypertext cleaning steps required to eliminate these violations are delegated from the data analysis systems, where they were handled traditionally, to the data gathering systems. We presented two algorithms for detecting templates: an algorithm that fits small document sets and an algorithm that fits large document sets. Both algorithms are highly efficient in terms of time and space, and thus can be run by a data gathering system on large collections of hypertext documents stored locally. We demonstrated the benefit of template elimination, by showing experimentally that it improves the precision of the search engine Clever at all levels of recall.

There are a number of open questions to be addressed in future work:

1. Show that the techniques proposed in this paper, and other techniques based on the Hypertext IR Principles improve the results of other common IR tasks, such as clustering, classification and focused crawling.
2. Find other general methods to perform data cleaning in hypertext corpora.
3. Find more detailed linguistic or statistical *query independent* methods for identifying and enumerating pagelets. We are currently using very naive heuristic methods.
4. Find other uses of templates. A few examples may be: facilitating spoken interfaces for the web, and compressing web pages for PDA and cell phone browsers.
5. Can templates be an indicator of authority or commercialization of web pages?

7. ACKNOWLEDGMENTS

We would like to thank Inbal Bar-Yossef for her invaluable assistance with running the experiments and evaluating their results. We thank Ravi Kumar and the anonymous referees for useful comments.

8. REFERENCES

[1] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In *Proceedings of the Twentieth International Conference on Very Large Databases*, pages 487–499, Santiago, Chile, 1994.

[2] K. Bharat and M. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 104–111, 1998.

[3] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the 7th International World Wide Web Conference (WWW7)*, pages 107–117, 1998.

Query	Fraction of Template Pages	Query	Fraction of Template Pages
affirmative action	45%	lyme disease	32%
alcoholism	42%	mutual funds	**67%**
amusement parks	43%	National parks	33%
architecture	**68%**	parallel architecture	21%
bicycling	49%	Penelope Fitzgerald	**68%**
blues	25%	recycling cans	**64%**
cheese	39%	rock climbing	40%
citrus groves	32%	San Francisco	**64%**
classical guitar	38%	Shakespeare	39%
computer vision	32%	stamp collecting	41%
cruises	46%	sushi	43%
Death Valley	51%	table tennis	44%
field hockey	54%	telecommuting	39%
gardening	56%	Thailand tourism	37%
graphic design	28%	vintage cars	22%
Gulf war	40%	volcano	48%
HIV	43%	zen buddhism	44%
Java	**62%**	Zener	13%
Lipari	53%		
Average	43 %		

Table 2: Fraction of base set pages that contain template pagelets. The results (an average of 43% of the pages contain templates) indicate how pervasive the template phenomenon has become.

[4] A. Z. Broder, S. C. Glassman, and M. S. Manasse. Syntactic clustering of the web. In *Proceedings of the 6th International World Wide Web Conference (WWW6)*, pages 1157–1166, 1997.

[5] V. Bush. As we may think. *The Atlantic Monthly*, 176(1):101–108, July 1945.

[6] S. Chakrabarti. Integrating the document object model with hyperlinks for enhanced topic distillation and information extraction. In *Proceedings of the 10th International World Wide Web Conference (WWW2001)*, pages 211–220, 2001.

[7] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, and S. Rajagopalan. Automatic resource list compilation by analyzing hyperlink structure and associated text. In *Proceedings of the 7th International World Wide Web Conference (WWW7)*, pages 65–74, 1998.

[8] S. Chakrabarti, B. Dom, D. Gibson, R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Topic distillation and spectral filtering. *Artificial Intelligence Review*, 13(5-6):409–435, 1999.

[9] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data*, pages 307–318, 1998.

[10] S. Chakrabarti, B. Dom, P. Raghavan, S. Rajagopalan, D. Gibson, and J. Kleinberg. Automatic resource compilation by analyzing hyperlink structure and associated text. In *Proceedings of the 7th International World Wide Web Conference (WWW7)*, pages 65–74, 1998.

[11] S. Chakrabarti, M. Joshi, and V. Tawde. Enhanced topic distillation using text, markup tags, and hyperlinks. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2001.

[12] S. Chakrabarti, M. van den Berg, and B. Dom. Distributed hypertext resource discovery through examples. In *Proceedings of the 25th International Conference on Very Large Databases (VLDB)*, pages 375–386, 1999.

[13] S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: A new approach to topic-specific web resource discovery. In *Proceedings of the 8th International World Wide Web Conference (WWW8)*, pages 1623–1640, 1999.

[14] B. D. Davison. Recognizing nepotistic links on the web. In *Proceedings of the AAAI-2000 Workshop on Artificial Intelligence for Web Search*, pages 23–28, 2000.

[15] J. Dean and M. Henzinger. Finding related pages in the world wide web. In *Proceedings of the 8th International World Wide Web Conference (WWW8)*, pages 1467–1479, 1999.

[16] E. Garfield. "Citation Analysis as a Tool in Journal Evaluation". *Science*, 178:471–479, 1972.

[17] Google. Google. `http://www.google.com`.

[18] M. Kessler. Bibliographic coupling between scientific papers. *American Documentation*, 14:10–25, 1963.

[19] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, pages 604–632, 1999.

[20] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Trawling the Web for emerging cyber-communities. In *Proceedings of the 8th International World Wide Web Conference (WWW8)*, pages 1481–1493, 1999.

[21] R. Lempel and S. Moran. The stochastic approach for link-structure analysis (SALSA) and the TKC effect. *Computer Networks (Amsterdam, Netherlands: 1999)*, 33(1–6):387–401, June 2000.

[22] Y. Maarek, D. Berry, and G. Kaiser. An information

retrieval approach for automatically constructing software libraries. *Transactions on Software Engineering*, 17(8):800–813, 1991.

[23] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Computer Science Department, Stanford University, 1998.

[24] G. Pinski and F. Narin. Citation influence for journal aggregates of scientific publications: Theory, with application to the literature of physics. *Inf. Proc. and Management*, 12, 1976.

[25] P. Pirolli, J. E. Pitkow, and R. Rao. Silk from a sow's ear: Extracting usable structures from the Web. In *Conference Proceedings on Human Factors and Computing (CHI)*, pages 118–125, 1996.

[26] H. Small. Co-citation in the scientific literature: A new measure of the relationship between two documents. *Journal of the American Society for Information Science*, 24:265–269, 1973.