

# A Modern Approach to Searching the World Wide Web: Ranking Pages by Inference over Content

Bronson Trevor<sup>1</sup>, Edgar Weippl<sup>2</sup>, Werner Winiwarter<sup>3</sup>

<sup>1</sup> Beloit College, Beloit,  
WI 53511, USA

trevorb@stu.beloit.edu

<sup>2</sup> Software Competence Center Hagenberg,  
A-4232 Hagenberg, Austria  
edgar.weippl@scch.at

<sup>3</sup> E-Commerce Competence Center,  
A-1070 Vienna, Austria  
werner.winiwarter@ec3.at

**Abstract.** The Hypertext-based Webs such as Intranets contain a vast amount of information pertaining to an enormous number of subjects. It is, however, an organically grown and thus essentially structureless environment that is in a constant state of flux. Therefore, finding useful information pertaining to a particular topic is oftentimes a difficult task. Search engines were designed with the intent of easing the burden on the individuals perusing the Web for specific topics. Traditionally, Web search engines have used straightforward—and relatively naïve—approaches towards indexing and ranking pages pertaining to a particular subject. As our understanding of hyperlinked environments has improved, algorithmic tools have been developed that more effectively distill the plethora of information that exists within this environment. We will briefly discuss the history of the World Wide Web, the approaches employed by “traditional” search engines, and how alternative techniques can improve upon older approaches. We find that new techniques build upon, rather than replace, previous approaches, and that the problem of searching the Web is one that evolves as our understanding of the Web’s structure improves.

## 1 Introduction

In today’s world of the World Wide Web, Gnutella, Java Server Pages and document databases, distinction among text, databases and documents gradually disappear. Digital documents are essential in day-to-day organizational work but yet retrieving and managing documents is an erratic and unsystematic endeavor.

Over the past decade, we have witnessed an exponential growth in the size of both the Internet and corporate Intranets. Much of that growth can be traced back to the inception of the World Wide Web in the late 1980’s. Since that time, the Internet, and more specifically the World Wide Web (or just ‘the Web’) has grown to encompass an extraordinarily vast number of documents pertaining to an enormous number of

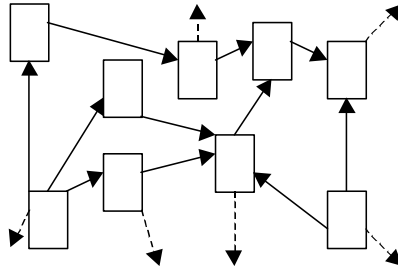
subjects. Moreover, over the past couple of years Intranets based on the same technology as the Web have had a large influence on all aspects of modern life, changing the way we think about communities, business, and the way we interact with the world at large.

Following the example set by the Internet, the paradigm of corporate Intranets is one of a non-centralized, collaborative environment in which (at an organizational level) no one page has explicit precedence over any other. However, we find that some pages do in fact have implicit authority over others in relation to particular subjects.

While the initial paradigm of hyperlinked environments was relatively focused and simple, the mere scale to which its size has grown presents a number of organizational problems. Kleinberg notes that the Web is a “hypertext corpus of enormous complexity that continues to expand at a phenomenal rate” [1]. See Figure 1 for a visual representation of a small chunk of the Web. It is an environment in which millions of participants are continually creating new hyperlinked content – some estimates claim that up to 40% of the Web changes every month [2]. This content consists of information relating to an extraordinarily diverse number of subjects, and even though an individual Web publisher can impose a local order on her own set of documents, the global organization of the Web is entirely unplanned [7]. This de-centralized model allows for the unrestricted ability to publish e-documents, but presents an enormous structural problem: the Web has no extrinsic order. Because of the lack of global organization, high-level structure can be found only by a post-analysis of pages on the Web [1].

In this paper, we examine a broad number of methodologies for organizing the publicly available pages on the Web. An overview of various algorithmic tools that have been developed to more effectively distill the plethora of information that exists on the Web is given. We will explore some older searching algorithms that can be applied to search the Web (simplifications of approaches possibly employed by various contemporary search engines), and explore how more modern techniques build upon older, more ‘traditional’ approaches. The final focus of this paper is a discussion of hyperlink analysis, which is one method for ranking Web pages in relation to a particular query. Other approaches are discussed to provide proper context and background for examining the advantages and pitfalls of this ranking method. This paper attempts to illustrate how a smart approach, such as hyperlink analysis, can aid in a post-analysis of the Web’s structure [1].

Understanding how contemporary search engines work is essential when designing an organization’s Intranet. Searching and retrieving documents can probably be considered to be the foundation of any attempt of implementing a knowledge management strategy.



**Fig. 1.** A small set of hyperlinked Web pages. The arrows represent hyperlinks, and the dashed arrows represent hyperlinks to pages not depicted.

## 2 Search Engines

Doing this post-analysis of the entire body of Web pages available on the Internet is a daunting task. A 1999 study estimated that the number of publicly accessible Web pages available at that time numbered about 800 million (about 6 terabytes of text data), on about 3 million discrete Web servers. Sorting through all of that data requires human intervention of some sort, either by discovering and sorting pages manually or through the implementation of an algorithm that will somehow help to find order among the chaos of the Web. A search engine is a utility that somehow manages an index of the Web, and presents end users with relevant pages pertaining to a user-supplied query. Presenting users with appropriate pages related to their query requires some sort of ranking scheme. The combination of these three elements, the query, the index, and the ranking scheme, forms the basic structure of all Web search engines.

### 2.1 The Query

One way to organize Web pages that allows for fast access to information regarding a particular subject is to sort them by topic. This way, a search engine can relate topical information gathered about the Web to a user-supplied query. Conceptually, this is a straightforward organizational scheme: a user will hopefully know the subject on which they want to find information. One could argue that a good search engine would take the input query and return a concise list of highly relevant Web pages, at least some of which are authorities [1] on the subject. Think of an authority as an of authoritative or official site pertaining to a topic. For example, one might say that the Porsche Automotive Group Home Page [3] is an authority on a query for 'Porsche.' Generally, a Web page is deemed an authority on subject  $s$  if it is particularly relevant to  $s$ .

Before a search engine can begin to apply a query to an index of pages, some form of automatic query refinement must take place. This might take the form of removing

certain globally redundant, unnecessary, or ambiguous words such as ‘the,’ ‘and,’ ‘it,’ etc. These words are commonly referred to as stopwords.

A query term can also have numerous senses. If a user enters a query such as “apple”, the search engine needs to somehow determine in what context the term is being used: it needs to ascertain whether to return pages relating to Apple Computer, Inc. [4] or the fruit. An informed search engine user will form queries in a specific and concise manner, but when designing a search engine it must be taken into account that not all users will be quite so deft. A good search engine will employ some method for dealing with such situations. At this point, a search engine applies the query to a database of pages, or index, and attempts to locate pages that are topically related.

## **2.2 Indexing Web Pages**

Once the search engine has parsed a user-supplied query and determined the exact terms to search for, it has to apply them somehow and come up with a list of results for presentation to the user. There are several ways of approaching this problem. Since “speed (i.e., search engine search and retrieval time plus communication delays) has consistently been cited as ‘the most commonly experienced problem on the Web’” [5], it is not practical for a search engine to retrieve and parse every single Web document on the fly. Thus search engines use a pre-collected list of pages (collected and maintained in different ways) called an index, which is searched in lieu of the alternative.

### **Manual Indexing**

One approach to the categorization of Web documents is called manual indexing. In this scheme, an individual or group of individuals maintain a human-readable list of Web documents that are categorized by hand—no computer-based algorithm is applied directly to categorization. It is also possible to allow for the user-submission of Web pages, negating to some degree the need for the maintainers of a manual index to search out pages, but in the end the placement of the submitted pages within the search engine’s index still relies on human judgment. Yahoo! [5] is one example of a search engine that uses manual indexing.

This method has the disadvantage of being rather labor-intensive. As the size of the Web grows by leaps and bounds, it is becoming more and more difficult to manually maintain a list of every discrete subject matter available. Digressing into a topic that is laid out in more detail in Section 2.3, ranking scheme, it might be said that manual indexing, although labor-intensive, has an advantage because the pages are ranked using active human judgment, and though far from perfect, “compared to most automatic indexers, human indexing is currently the most accurate because experts on popular subjects organize and compile the directories in a way which they believe facilitates the search process” [3]. As automatic algorithmic-based indexing schemes improve, the need for search engines that employ manual indexing is slowly eclipsed, and thus, “manual indexing is likely to become obsolete over the long term” [3].

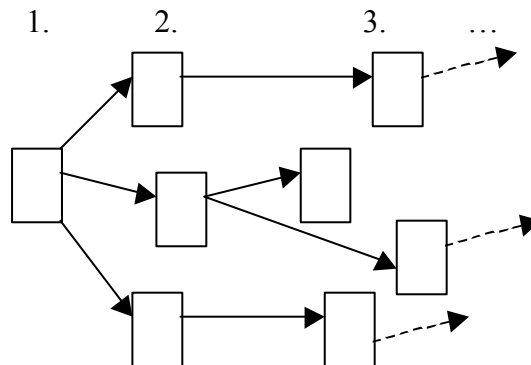
### Robots, Crawlers, Spiders, et al.

Another way of creating an index of available Web pages is through an automatic Web page discovery and retrieval system. A Search Engine can employ an application that “automatically traverses the Web’s hypertext structure by retrieving a document, and recursively retrieving all documents that are referenced” [7]. Note that said application can take on many different forms, hence “they are most commonly referred to as crawlers, but are also known as ants, automatic indexers, bots, spiders, Web robots, and worms” [3]. They fall under the more general category of Intelligent Agents, which are programs that employ algorithmic analysis in an attempt to recreate some form of human judgment (Intelligent Agents fall beyond the scope of this paper).

An application that automatically traverses the Web might work as follows:

- A base set of known working hyperlinks is used as a starting point.
- Each of these hyperlinks is placed in a queue. The Web robot will endeavor to retrieve each of the pages in this queue. Because the Web is a hyperlinked environment, each of these pages will in all likelihood have links to various other pages on various other Web servers.
- The Web robot retrieves the first page in its queue and adds an entry for this page in the index.
- Then, it adds each of the hyperlinks that exist on that Web page to the queue.
- It repeats this process of retrieval and recording for each page in its queue.

Thus, for each page discovered, the list of pages that remain to be gathered will (ideally) grow, and so will the index. Indeed, it becomes evident that a Web robot’s indexing process is one that builds upon itself: the more pages it retrieves, the more pages it discovers that need to be added to its index. See Figure 2 for a visual representation of this process.



**Fig. 2.** Discovering 7 pages from 1 initial page, moving left to right. The first three steps are depicted, the three dashed arrows at the end all point to further pages with more hyperlinks.

This process of automatic page discovery takes a great burden off of a search engines maintenance team. Rather than having to rely on announcements and user-submissions to grow the index, the indexing application is constantly and automatically discovering

new pages for inclusion in the index. Day-to-day maintenance of the index is aided greatly by the use of such an algorithm, but skilled programmers are still required to maintain and update the actual indexing application, as opposed to manual indexing where the human labor is focused on directly maintaining the index.

To be effective, the implementation of a Web-traversal algorithm needs to be done at a rather large scale. Even with an automated system such as this, indexing a significant portion of the Web is a gigantic task. Imagine the indexing application were able to discover 1 million new pages a day (about 42,000 an hour or 700 per minute—possibly a generous estimate); it would take a bit over 2 years to index a static, non-changing body of 800,000,000 pages. Given that the turnover/change rate of the Web is significant—recall it is estimated to be around 40% [3]—it becomes ever more difficult to maintain an accurate picture of the Web, even with automated tools such as a Web spider. Many commercial search engines utilize variations on this approach to indexing.

Such a demanding application also requires a fair amount of data transmission bandwidth simply for the indexing of Web pages. The implementation of a scaleable and effective manual or automatic indexing scheme will almost certainly take on a large scale. There is an alternative: a scheme which makes no use of a persistent, local index.

#### **Metasearching: parallel queries in lieu of a static index**

As an alternative to maintaining an on-site index of Web pages, search engines exist which take advantage of the fact that numerous other agencies have already done the bulk of the indexing for them. In lieu of keeping up a local Web index, these “meta” or “parallel” search engines automatically query various commercial search engines for every user query that is received. Thus, for every user-supplied query, this parallel search process consists of:

- A number of parallel queries to various search engines that maintain a static index.
- The retrieval of these results.
- The combination and ranking of these results for presentation to the user.

The idea is that by combining the results of various search engines, the Web pages presented to the user as relevant will hopefully represent a broader portion of the Internet than if just one search engine were to be used. Infind [3] is an example of a search engine that uses this method.

Given that an individual “search engine would have indexed between 3% and 34% of the possible total number of Web pages” [3] [5], such a parallel method has distinct advantages. That same year, it was estimated that the extent of overlap among the top six search engines was small, and their collective coverage was around 60% [3] [5]. A lack of significant overlap among different search engines means that results returned from discrete search engines are more likely to be distinct. A parallel search engine will have greater coverage by collecting and summarizing these distinct results. A follow-up study in Feb, 1999 showed that the situation of search engine coverage is only getting worse, “a far smaller proportion of the Web is now [1999] indexed with no engine covering more than 16% of the Web” [3] [9].

## 2.3 Ranking Scheme

Creating an index of the Web is only a step towards the implementation of a fast and accurate search engine. Once a search engine has an index of pages to work with, it has to be able to rank these pages in order of precedence for a specific query. A quick and informal check of AltaVista [10], a major search engine, showed that a search for the term 'Porsche' returned 830,938 relevant pages. Clearly, a user would be unable to peruse through all of these results; one study showed that most users become frustrated when they have to look through more than 2 or 3 pages of results [3]. In order to make sense of the mountain of relevant pages to a given query, some sort of ranking scheme must be employed so that the user will see the most relevant pages at the top of the result list. Kleinberg [1] notes that the idea of relevance has an inherent subjectivity; indeed, the notion is one that relies on human judgment and analysis. It becomes necessary to formulize and formalize in some fashion the idea of relevance and to extract from Web pages elements that somehow convey an implicit declaration or conferral of authority in the context of a particular query. There are a number of different ways to go about doing this, bearing in mind that "detailed information regarding ranking algorithms used by major search engines is not publicly available" [3] because of their proprietary nature.

### **Boolean modeling**

A simple use of vectors for the ranking of Web pages is employed in the Boolean ranking scheme. Each page  $p$  is thought of as a collection of  $n$  distinct words, with an entry for each of these  $n$  distinct words in the vector that represents  $p$ . The query is then compared to the vector for each page that exists in the index. If the entire Boolean query evaluates to true for a page  $p$ , then  $p$  is deemed relevant by this ranking scheme. That is to say, if all three of the terms 'Porsche', 'motor', and 'cars' appear in  $p$ , then  $p$  is relevant to the aforementioned query. So, we find that the Boolean model "considers that index terms are either present or absent in a document" [11].

It is a simple and efficient scheme, and can work well with a small body of documents and verbose, explicit queries; the Boolean format allows for a user to form very specific queries. It has the drawback, however, of being so absolute, indeed, "there is no notion of a partial match to the query conditions" [11]. Without this idea of a partial match, the Boolean ranking scheme breaks down under conditions where the query is vague or there exists an enormous number of documents to search. A Boolean ranking scheme might determine that 830,938 pages are relevant to a query, but they would all be equally relevant. Thus, we find that "many refinements of the Boolean model exist" [3]. In its basic form, the Boolean ranking model predates more complex models, and, in reality, has been eclipsed by them.

### **Vector-space modeling**

Vector-space modeling [12] is a methodology employed by at least some of the major search engines [3]. This is a method that computes a similarity measure among a set of Web pages. Like Boolean modeling, the vector-space method views each page as a collection of  $n$  words [13]. This set is made up of the distinct words that appear on the

page. Each word is given a score relating to the number of times that it appears on the page. For example, if the term ‘Porsche’ appears 7 times on page  $p$ , and the word ‘car’ appears 50 times on  $p$ , ‘Porsche’ would have a 7 associated with it and ‘car’ a 50 for  $p$ .

$$\langle \dots, \text{car}=50, \dots, \text{Porsche}=7, \dots \rangle$$

This reduces  $p$  to an  $n$ -dimensional vector:

$$\langle w_1, \dots, w_i, w_{i+1}, \dots, w_n \rangle$$

with the  $w_i$  term representing the score for word  $i$ ,  $w_1$  being the highest ranked word on  $p$  and  $w_n$  being the lowest [13].

The next step to understanding this method is to understand the idea of a word score. One way of computing the word score of a word  $i$  is accomplished by multiplying the number of instances of  $i$  in the document times the inverse of the number of times that  $i$  appears in the search engines index of pages. For example, imagine the term ‘Porsche’ appears 7 times on  $p$  and 14,420 times in a particular search engines index, the equation:

$$7 \times \frac{1}{14,420} = 0.000485$$

would calculate the word score for ‘Porsche’ for this particular  $p$  to be 0.000485. Indeed, it becomes evident that an uncommon term, such as ‘Porsche’ will receive a higher score than a very common word, such as ‘car’, because it will appear much less frequently in the search engines index of the Web. Even if the word car appears 50 times on  $p$ , it occurs many more times in the index, say, 4,200,000.

$$50 \times \frac{1}{4,200,000} \approx 0.0000119$$

Thus, uncommon, unique terms are not outweighed by common but important terms. Empirically, we find that common terms refer to general ideas, such as ‘car,’ and uncommon terms refer to specific ideas, such as ‘Porsche.’ Note that the use of this particular scoring methodology is just one variation on the vector-space model.

Now that word scores are calculated, we need to relate the query to the index. “The similarity between query  $q$  and document  $p$  can then be defined as the inner product of the  $q$  and  $p$  vectors” [13]:  $q$  is first projected onto a pseudopage, i.e., it is mapped onto a temporary “page” that is ranked using the same methodology as the pages in the search engines index. Common terms are outweighed by uncommon terms. For instance, a query for “Porsche motor cars” is a query specifically for ‘Porsche,’ not generally for ‘motor’ or ‘car.’ This way, a pseudopage (which is just a representation of the original query) can be easily ranked alongside other real pages, and the ones that rank closely to the pseudopage can be deemed as relevant to the query [3]. A pseudopage with a high word score for ‘Porsche’ will be ranked closely to pages with a high word score for ‘Porsche.’ Thus, it becomes evident that processing the query takes place in two steps: projection and mapping. The query is projected onto a pseudopage, and the pseudopage is placed, or mapped, among the search engines index of

pages. The pages that end up closest to the pseudopage in this map are the most relevant pages in the index.

This ranking scheme certainly approaches a formulation of the idea of relevance [1] between documents, but has significant drawbacks. It is ranking pages solely on content, and there is no guarantee that a page will be ranked well if content is the only factor taken into consideration. Kleinberg [1] uses the example query of ‘search engines’ to illustrate that a pure vector-space ranking scheme will likely miss the home pages of the major search engines simply because the term “search engine” appears nowhere in the text body of those pages.

This ranking scheme also raises the question: does the word  $q$  always appear many times on a page that is relevant to  $q$ ? Probably not, thus we see that the vector-space model also enables Web designers to employ a devious method referred to as spamming or Web page persuasion. Spamming is the “excessive, repeated use of key words or ‘hidden’ text purposely inserted into a Web page to promote retrieval by search engines” [3]. If a Web page designer wanted to promote a page about transmission repair, inserting the phrase ‘transmission repair’ many times as hidden text into a Web document would cause a vector-space ranking algorithm to rank this page higher for a query for ‘transmission repair.’ A similar problem is encountered with the use of metadata, discussed in section 2.3.4.

#### **Latent semantic indexing and the ideas of synonymy and polysemy**

Latent Semantic indexing (or LSI) [14] “one of the more widely used vector space model-based algorithms,” [3] is a ranking method that is similar to the previously described vector-space method. It ranks pages by content, but not just by simple pattern matching. Instead of merely ranking words on a page, LSI takes into account the ideas of synonymy and polysemy [3]. Synonymy refers to the notion that two ideas or objects are in some way alike or synonymous, polysemy refers to the notion that a single term might be ambiguous if not contextualized properly, i.e., that a term might have more than one discrete sense or meaning. A synonymy exists between the terms ‘car’ and ‘automobile,’ a polysemy exists between the various senses of the word “transmission.” By taking into account these two ideas, a LSI system can more effectively group similar pages together, as well as distinguish between pages that use similar terms, such as ‘apple’ and ‘Apple Computer, Inc.’

LSI will automatically discover, using statistical methods, terms that are often used in the same context [15]. For example, an LSI application indexing the Web would likely find that pages using the terms ‘Apple’ and ‘computer’ will also often contain the term ‘MacOS,’ and that the term ‘Microsoft’ is often used in conjunction with ‘Windows.’ Note that this is not necessarily a two-way relationship: the term ‘windows’ certainly appears in many documents that are not necessarily related to ‘Microsoft.’

This approach remains language independent; the algorithm will notice the same type of synonymy between the French terms ‘Peugeot’ and ‘voiture.’ Thus, LSI comes in useful when “there is a need to retrieve information in multiple languages without requiring translation of queries or documents” [15]. It becomes evident that LSI is an evolution of the previously described ranking methods; it builds on the idea that it is

possible to extract an idea of document relevance without any explicit or stated knowledge of a Web page's content.

### **Metadata**

Metadata, not to be confused with metasearch engines, is a concept that was fostered by the idea that an explicit and concise statement about a Web page's content should aid in the ranking of pages for a particular query. "In the context of Web pages on the Internet, the term 'metadata' usually refers to an invisible file attached to a Web page that facilitates collection of information by automatic indexers" [3]. An interesting concept, it has definite advantages but falls apart in the unregulated atmosphere of the Internet.

This concept relies on several factors. Firstly, some sort of standard needs to be drawn out so that any search engine that retrieves a page  $p$  will be able to properly recognize any metadata that might be associated with  $p$ . Several standards have been proposed, one of them is the Dublin Core Metadata standard [3] [16] which describes a number of elements that should be present in order to facilitate page ranking. There are 15 elements, including date, title, creator, subject, description, and language among others. Since page properties are explicitly stated by a pages' associated metadata, the burden of guesswork on a search engine is greatly reduced. Properly formatted metadata would greatly aid in the grouping of relevant pages.

We see that the effective use of metadata relies on acceptance of a standard and the proper, honest use of the metadata by Web page designers. A standard is not worth the paper it is printed on (or the Web space it takes up) if only a few Web designers actually use it. Further, if the Web page designers can use the metadata towards some dubious purpose (along the lines of spamming) then the acceptance of such a standard could actually hinder the accuracy of search engines in the long run. Today, despite several proposals, no overriding standard for the use of metadata as described here has been put into widespread use [3].

## **3. Analysis of hyperlink structure**

A common thread runs through all of the aforementioned ranking schemes: they all rely in some way on the actual textual content of the page (or its associated metadata) for ranking the page. What they fail to take into account is that the Web is more than just a simple collection of documents, it is a collection of hyperlinked documents that are thickly laden with reference upon reference to other yet other documents on the Web. Recall Figure 1 for a visualization of this idea. A Web spider (Section 2.2.2) builds on this idea for the indexing of Web pages; by inference it seems plausible that it might be possible to rank pages taking the Web's greater structure into account—no one page depends on the rest of the Web, but no page is effectively independent of it either. In section 3 we will explore one take on this idea of ranking pages by inference over content, the algorithm laid out by Jon Kleinberg [1].

### 3.1. Authorities and hubs

When Web page designers make a hyperlink  $h$  on page  $p$ , they are using their judgment to confer that that by following  $h$ , a user will end up at some page  $q$  that relates, in some way, shape, or form, to the topic of  $p$ . A Volkswagen enthusiast might design a Web page  $v$  about Volkswagen automobiles, and it would stand to reason that outgoing links from this page would likely relate to the content of this page. They might, for example, place a link on  $v$  to the Volkswagen home page  $p$  <www.volkswagen.com> [17], or to their favorite car parts supplier  $s$ . That parts supplier might, for the purposes of this simple example, also have a link to  $p$  somewhere on  $s$ . Figure 3 illustrates this example.

Assume that  $v$  is a source of information on Volkswagen automobiles. We can infer that  $s$  and  $p$  are also somehow topically related. Then, take note of the fact that of the three pages,  $p$  has the largest number of incoming links (2 in this very simple example.) Thus, the claim could be made that  $p$  is an authority on Volkswagen automobiles; other pages that are topically related seem to point back to  $p$ , the Volkswagen Corporation home page.

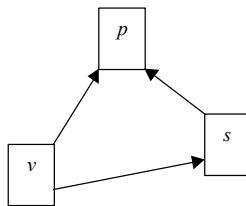


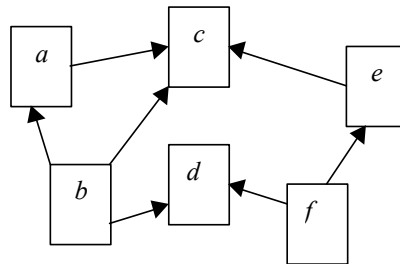
Fig. 3.  $v$  and  $s$  bestowing authority on page  $p$

A hub is a Web page that contains links to many pages relevant to a particular subject. In the example above,  $v$  would be designated a hub page (in relation to the subject) because it contains the greatest number of outgoing links to pages relevant to VW. Thus it becomes apparent that the two notions are interdependent: a good authority should be pointed to by many good hubs, a good hub should point to many good authorities.

### 3.2. Focused Subgraphs of the Web

An assumption made in the previous example was that we were somehow able to determine the initial condition that  $v$  was relevant to Volkswagen. Because of the circular nature of the authority-hub relationship, we must be able to guarantee that this algorithm looks only at pages related (loosely or specifically) to the query, and analyze the hyperlink structure within this set of pages. For example, if we are given a set of pages that are known to be somehow related to Volkswagen automobiles, we could then determine their internal link structure and find the best authorities and hubs within that set of pages.

Figure 3 is an example of a small focused subgraph of the Web. It is a subgraph because it contains 3 pages out of the entire set of Web pages on the Internet, it is focused because all the pages relate in one way or another to the same subject. Consider a more complex graph presented in Figure 4.



**Fig. 4.** A focused subgraph (6 pages) of the Web. Note that *c* has the largest number of incoming links, and that *b* has the largest number of outgoing links

If we are given the precondition that all of the pages in the graph above relate to query *q*, then we can infer that *c* is an authority on *q* and *b* is a hub page for *q*. We use a variation of metasearching (Section 2.2.3) to obtain the initial set of pages, which we then use to build a subgraph and analyze link structure. Before we can begin our analysis, we query one existing, content-based ranking search engine and retrieve an initial set of results. We disregard the order in which the queried engine returns the results, we are only interested in ensuring that we work within a focused group of relevant pages. Kleinberg suggests the use of AltaVista [10].

This is only the first step, we in fact want to grow this initial set of pages to include pages that would not necessarily have been returned by the content-based search engine that we queried. We grow this initial set of pages by the following steps:

- For each page *p* in our initial set:
- Store each outgoing link from *p* in the extended set.
- Find *n* pages pointing to *p* and add hyperlinks to them to the extended set.

We find the *n* pages that point to *p* by querying a search engine that will locate incoming links to a page (Kleinberg suggests setting *n* to 50.) Thus our initial set is grown into the extended set by retrieving out-links and in-links that point from and to members of the set. If the hyperlink-relevance hypothesis is correct, then these pages in the extended set should all be more or less related to the initial query, and the extended set should contain a number of authorities and hubs to discover by hyperlink analysis.

### 3.3 Hyperlink analysis

Having collected a set of pages known to be relevant to the query, it is now possible to begin analyzing the data that we have collected. Remember that the authority-

hub relationship is one that is circular and self-reinforcing. A good authority is pointed to by good hubs, and a good hub points to a lot of good authorities. To break this circularity we use an iterative approach to analyzing the pages in our focused subgraph. We want to keep track of two things for each hyperlink in our set:

- How good of an authority is it?
- How good of a hub is it?

It is certainly possible that a page could be both. We associate a relative weight with each page, the pages with the highest weights in each category after the algorithm has been run will be deemed the best authorities and hubs in our collection.

- We state that for each authority  $p$ , its authority weight  $x$  is the sum of all of the hub weights  $\{y_1, \dots, y_n\}$  of the  $\{q\}$  pages pointing to  $p$ .
- For each hub  $q$ , its hub weight  $y$  is the sum of all the authority weights  $\{x_1, \dots, x_n\}$  of the  $\{p\}$  pages that  $q$  points to.

A simplified version of the basic algorithm used to calculate these weights works in the following iterative fashion:

- Set all weights to 1.
- Repeat  $k$  times:
  - Calculate authority weights for each page in our subgraph.
  - Calculate hub weights for each page in our subgraph.
  - Normalize all weights.

Kleinberg uses empirical data to state that 20 is a good value for  $k$ . We normalize the weights so that they all fall between 0 and 1. This way, the weights remain the same in relation to one another, and converge on certain values depending on the specific subgraph. One can see that after one iteration, pages that have many in-links from members of the set will begin to take on higher authority weights, and pages that have many out-links to members of the set will begin to take on higher hub weights. With each successive pass of the algorithm, the pages that are pointed to by the best hubs become better authorities, and the pages that point to the best authorities become better hubs.

After running this algorithm, each page will have associated with it two appropriate relative weights. We can choose the top  $c$  ranked pages for each of the two weights and present them to the user. A discussion of various implementations of this algorithm is presented in Section 3.5, in which we discover that Kleinberg's approach, while quite effective, robust, and original, does have certain drawbacks.

### 3.4. Implementations of Hyperlink Analysis

The idea of hyperlink analysis has been used in the implementation of several Web search engines in use today. One such search engine is Google.com [18]. The Google search engine uses an optimized implementation of Kleinberg's idea. In a strict implementation, Kleinberg's algorithm uses no local index—for each query we 'piggy-back' onto AltaVista and use its result list to build our subgraph. To effectively use Kleinberg's algorithm in a real-world application, the designers of Google.com opted for the use of a local index (gathered by crawling), which is searched in lieu of the

piggyback step. This yields much faster search times because we are accessing a local index instead of relying on another search engine [19].

Another search engine utilizing a variation on Kleinberg's approach is Infind [8]. A piggyback step is still used, but instead of just querying AltaVista, a meta-searching approach is used. By querying several different search engines in the initial step, a larger variety of the Web is included into the subgraph. Thus, in theory, a better result set is obtained.

Finally, a version of Kleinberg's algorithm was implemented to empirically study search results. Java 1.3 was chosen, as it includes a number of useful methods for utilizing http as well as parsing the Web pages after they have been retrieved. This implementation led to a number of interesting observations. It was made painfully clear that Kleinberg's algorithm is not one optimized for speed, rather, it is a robust approach that sacrifices speed for a result set of high quality.

For every query, a strict implementation of Kleinberg's algorithm does 200 http requests, and then for each of these requests has to parse the page returned and do another http request to find 50 incoming links to a page. Thus, for every query we are required to retrieve 400 distinct Web pages, as well as the initial query to AltaVista. We then have to efficiently store all of the hyperlinks we find, and build the subgraph of the pages referenced. In this implementation, the subgraph-building step ended up being the most time consuming. Basically, the approach ended up using an optimized  $O(n^2)$  loop. With 1,000-10,000 pages in an average subgraph, a considerable amount of time was spent on this simple step.

#### **4. Conclusion**

Like the Web, the problem of searching the Web is enormous and complex. It is difficult to find an explicit order in a body of pages that exist in an essentially unstructured environment. As the statistics mentioned in Section 2 indicate, the methods currently used to index and rank Web pages are still in a premature state. Through exploring alternative approaches such as hyperlink analysis, our understanding of the Web's structure improves, leading to further refinements and progress towards the ideal: a search engine that indexes and ranks pages perfectly. For now, the first step is to define perfection.

#### **References**

1. Kleinberg, J. Authoritative Sources in a Hyperlinked Environment. 1998.
2. Kobayashi, M. & Takeda, K. Information Retrieval on the Web. 2000. ACM Computing Surveys, Vol. 32, No. 2, June 2000, 146-151, 153, 154, 155, 160.
3. Porsche AG Home Page. <http://www.porsche.com> (last visited 5/29/01)
4. Apple Computer, Inc. <http://www.apple.com> (last visited 5/29/01)

5. Lawrence, S. and Giles, C. 1998. Searching the world wide web. *Science* 280, 98-100.
6. Yahoo! Inc. <http://www.yahoo.com> (last visited 5/29/01)
7. The Web Robots FAQ. <http://info.webcrawler.com/mak/projects/robots/faq.html> (last visited 5/29/01)
8. InFind.com. <http://www.infind.com>
9. Lawrence, S. and Giles, C. 1999. Accessibility of information on the web, *Nature* 400, 107-109.
10. AltaVista. <http://www.altavista.com> (last visited 5/29/01)
11. Baeza-Yates, R. and Ribeiro-Neto, B. 1999. *Modern Information Retrieval*. Addison-Wesley, Reading, Ma. 26, 27.
12. Salton, G., Ed. 1971. *The Smart Retrieval System: Experiments in Automatic Document Processing*. Prentice-Hall, Englewood Cliffs, NJ.
13. An Efficient Algorithm To Rank Web Resources. <http://www9.org/w9cdrom/251/251.html> (last visited 5/29/01)
14. Deerwater, S., Dumai, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. 1990. Indexing by latent semantic analysis. *J. Am. Soc. Inf. Sci.* 41, 6, 391-407.
15. Telcordia™ Latent Semantic Indexing Software (LSI): Beyond Keyword Retrieval. <http://lsi.research.telcordia.com/lsi/papers/execsum.html> (last visited 5/29/01)
16. Dublin Core Metadata Standard Home Page. <http://dublincore.org/> (last visited 5/29/01)
17. Volkswagen of America, Inc. <http://www.volkswagen.com> (last visited 5/29/01)
18. Google.com. <http://www.google.com> (last visited 5/29/01)
19. Brinn, S., Page, L. The Anatomy of a Large-Scale Hypertextual Web Search Engine. [www7.scu.edu.au/programme/fullpapers/1921/com1921.htm](http://www7.scu.edu.au/programme/fullpapers/1921/com1921.htm) (last visited 5/29/01)