Usage Meets Link Analysis: Towards Improving Intranet and Site Specific Search via Usage Statistics

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Bilgehan Uygar Oztekin

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Prof. Vipin Kumar, Adviser

April 2005

Acknowledgments

First, I would like to express gratitude to my academic adviser, Prof. Vipin Kumar, for his continuous encouragement, motivation, patience, and financial support throughout my graduate studies; without his support and guidance, this thesis would not be possible. I would like to thank Prof. George Karypis, for guiding me during various projects and for providing a novel perspective whenever I needed. I would also like to thank Prof. Jaideep Srivastava and Prof. Gediminas Adomavicius, for their willingness to serve on my Ph.D. committee as well as their valuable advice and suggestions.

The Computer Science Department and Army High Performance Computing Research Center at the University of Minnesota have provided a wonderful research environment. I wish to thank many of my fellow colleagues, especially Dr. Eui-Hong Han, Levent Ertoz, Eric Eilertson, Aysel Ozgur, and Dr. Pang-Ning Tan who provided valuable feedback and support in many occasions.

I would also like to extend my sincere thanks to all my friends who have assisted me in so many ways during my studies at the University of Minnesota. Special thanks to Ismail Guler, Emin Aklik, Bayram Tekin, Aysel Ozgur, Levent Ertoz, Irene Moulitsas, Cetin Urtis, Pinar Tutus, Mohac Tekmen, Pinar Tekmen, and Emre Celebi for making my journey through graduate school a wonderful experience.

Lastly, I owe special gratitude to my parents, Dr. Erbil and Ayse Oztekin, and my extended family for offering me examples to look up to and for providing valuable perspectives, guidance, and support throughout my life. I would especially like to thank my aunt Dr. Fatos Vural, and my uncles Dr. Faruk Yarman, Dr. Siddik Yarman, and Dr. Tolga Yarman, with whom I had countless hours of discussions and joy, which, I believe, played a crucial role in my formation. I would also like to thank my sisters Ozde and Ilke, and my cousins Dervis Vural and Evren Yarman, who are also pursuing graduate studies in this part of the globe, for providing me a closer support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, link analysis, PageRank algorithm in particular, played a crucial role in improving Web scale search, and became one of the key components in ranking pages in leading search engines such as Google. Most of the work in the area of link analysis relies primarily on the static hyperlink structure, and to a lesser degree on the content of the pages. Both the link structure and the content of the pages are provided by the authors of the pages. In some sense, almost all previous link analysis approaches sees the world from the author's or site administrator's point of view. Another perspective that is often ignored is the users' perspective and preferences while traversing the pages/links. Intuitively, pages that are visited frequently or links that are traversed heavily are more important than the ones visited/traversed rarely. However, most of the traditional link analysis approaches make no or very little distinction between the two. Often, all pages/links are treated equally, or weighted via simple heuristics such as position and approximated visibility of the link within the page, or similarity of the anchor text to the query in the case of focused crawlers.

Search within a site, an intranet, or a subset of a site is quite different than Web scale search, and is important in various contexts. Many organizations including businesses, and governmental and educational institutions have substantial amount of information behind firewalls, inaccessible to general purpose search engines. Some services such as digital libraries, although transparently available to a subset of the users on the Internet, are not available to general public (IP based access restrictions, password protected portions etc.). There are also subsets of sites disallowed to web crawlers via the Robots Exclusion Protocol (robots.txt) [27]. These portions may still be important internally and worth indexing and searching within the organization. Specific search capabilities for a particular domain or subset of a site may also be desirable even though the pages may be publicly available and indexed by general purpose engines (timely updates, focused/compact search, additional capabilities, different vocabulary/term weighting, alternative ranking methods etc.). The need for a reliable searching method for these situations is an important need for the users of these pages, especially when the portion of interest is not accessible by Web scale search engines. The solution is often a site specific search engine, accompanied by a site level portal, both maintained by the organization itself.

Link analysis has been a major component in Web scale search. However, due to unavailability of global scale statistics such as linkage information and global scores, application of link analysis techniques is often less accurate on a site level. This is perhaps a minor issue for intranet search, in which pages are rarely linked from outside, but it is an important issue for searching within a domain or subset of a domain where substantial interaction from pages inside and outside may exist, suggesting that global information may offer a more accurate view. A related problem shared by both intranet and site specific search is that, on a site level or within an intranet, trivial and effective global scale filtering approaches (e.g. deemphasizing links within the site vs. links across sites, running the algorithm first on domain level, weighting the importance of domains, and then focusing on the page level etc.) are not easily applicable. On the other hand, site specific or intranet search engines have an important information source that is not readily available to Web scale search engines: Extensive usage information on the site level. This information can be easily extracted from usage logs and can be used to build a usage based graph in conjunction with the static linkage graph. Both link and page hit statistics, as well as statistics on how many different IPs (a crude identifier for unique users or user groups) has accessed a particular link/page in a given time window can be obtained. This information can complement the link structure, acting as a valuable signal and filtering mechanism.

Incorporation of usage information into link analysis and the overall ranking methods has another potential benefit: increased resistance against spamming. During the early days of Web search engines, one could boost the score of a page by modifying the page alone (adding/removing keywords, modifying the frequencies). In recent search engines using link analysis, in order to boost the score of a page considerably, one needs to boost the link structure around that page, requiring modification of a number of pages as opposed to a single page. Hence, link analysis is considered to offer relative resistance against spamming or voluntary/involuntary perturbations, compared to early search engines. If usage augmented link analysis algorithms are used, not only pages/links must be created/modified around the page that one desires to boost, but also, these pages/links should have relatively high usage, potentially offering an additional, valuable signal against spamming. The other side of the coin is that, by introducing a new type of signal (link analysis), a new door that can be abused has been opened (link based spamming). Similarly, adding usage information as a signal, opens up a new, potential way of perturbing the results (usage based spamming). Note that spamming is still a common problem in search engines. For every refinement or signal that one may introduce, new ways of abusing the scoring mechanism can be found and exploited. The aim, in general, is to make it increasingly difficult and expensive to achieve a significant effect, reducing the incentive for spamming. Usage information can be one of the valuable signals in this continuous process.

Intuitively, as well as supported by a study by Fagin et al. [13], site specific and intranet search is relatively spam free, as there is no compelling incentive for perturbing the results within a single organization. For this particular domain, usage information may offer a reliable signal to improve the ranking quality without concerns for usage based spamming. This is particularly important when no global linkage information is available, or when the corpus is poorly linked compared to the Web. This thesis proposes and investigates a number of approaches incorporating usage statistics for site specific and intranet search. Some of these techniques are directly usage based, and some incorporate usage statistics into leading link analysis approaches: PageRank, and HITS. One of the ranking methods in particular, Usage Aware PageRank, is promising and has a number of desirable features (discussed in Section 2.3.1).

The remainder of this chapter is organized as follows: Section 1.1 gives a brief history of the Internet and Web scale search engines, Section 1.2 provides selected related work in the area of link analysis and usage statistics, and finally, Section 1.3 summarizes the overall layout of the remaining chapters.

1.1 Internet and Web Search Engines, a Brief History

Roots of the Internet can be traced back to early 60s. Table 1.1 provides a brief overview of some of the important events and milestones in the history of the Internet and search engines (sources: [46], [8], [53]). One of the first papers on packet switching theory has been published in 1961 by Kleinrock [26]. The Transmission Control Protocol (TCP) is first introduced in 1974, and, later, revised to form the TCP/IP protocol, making it feasible to connect various disconnected networks that were being formed all around the World. The ARPANET, developed in the late 60s and early 70s, later on switching to the TCP/IP protocol, gradually connected different networks, forming the basis of the current Internet. In the early 70s, ARPANET had only a few nodes/hosts. However, this number steadily increased to a few dozens, and then, to hundreds. As a result of this constant expansion, finding hosts was becoming increasingly difficult. This problem is addressed by the introduction of Domain Name System (DNS) in 1974, organizing machines into domains and mapping host names onto IP addresses. Various protocols that are integral parts of the Internet

60s		Early developments in networking
	1961	First paper on packet switching theory
	1965	ARPA sponsors study on cooperative network of time sharing computers
	1969	ARPANET is commissioned by DoD for networking research
70s		ARPANET, limited number of nodes
	1970	AlohaNet, the first packet based wireless network is operational
		First cross-country link connects UCLA and BBN at 56kbps
	1971	15 nodes and 23 hosts, email is invented
	1972	Telnet specification. France is building counterpart of ARPANET
	1973	File Transfer Protocol (FTP) specification
	1974	Transmission Control Protocol (TCP) specification
	1975	First ARPANET mailing list, satellite links across two oceans
	1977	Mail specification
	1978	USENET (news groups) is established
80s		Continuous expansion, transition to TCP/IP, merging of networks
	1980	ARPANET grinds to a halt due to an accidental virus
	1982	Norway connects through TCP/IP
		EUnet provides email and USENET services in parts of Europe
	1984	1,000 hosts, Domain Name System (DNS)
	1987	10,000 hosts
	1988	Internet Relay Chat (IRC)
		Regional networks continues to appear and join the Internet
	1989	100,000 hosts
Early 90s		Transition to WWW, countries/networks continue to join, exponential growth, need for search engines
	1990	ARPANET ceases to exist. First commercial dial-up ISP is operational.
	1991	World Wide Web by CERN, Gopher by University of Minnesota
		PGP (Pretty Good Privacy) released by Zimmerman.
	1992	1,000,000 hosts
	1993	Mosaic, the first widespread Web browser, becomes immediately popular.
		WWW Wonderer, first widely acclaimed Web robot
	1994	Galaxy, the first searchable directory. Lycos indexes 54K documents.
Late 90s		Search engine boom
	1995	Infoseek(Feb) and Metacrawler(June) launched.
		Altavista(Dec) is launched and has instant popularity.
	1996	Inktomi founded. Hotbot claims to have the ability to index 10M pages/day
	1997	Ask Jeeves, Northern Light
	1998	Open Directory Project, Page and Brin introduces Google(Sep), MSN search
		Directhit uses click information in ranking.
	1999	Fast search indexes 200M Web pages.

Table 1.1. Selected events and milestones in the history of the Internet and Web search engines

experience today were also introduced in 70s (Telnet, FTP, Mail, USENET).

By mid-80s the number hosts reached 1000; by late 80s, it reached 100,000; and by early 90s, the number of estimated hosts broke 1,000,000. The number of local networks all around the world steadily increased as well during this period. A portion of these networks, joined the network of networks, forming the basis of "the Internet" as we know today. In late 80s and early 90s, the expansion is also visible in terms of number of nations connected via the Internet, spanning majority of the world by late 90s.

In 1991, two alternative protocols for browsing resources and information were introduced: Text based Gopher protocol [2] by University of Minnesota, and the World Wide Web (WWW) protocol, including support for images and visual elements, by CERN. Mosaic, the first widespread WWW browser is launched in 1993 and became extremely popular. As the number of users having access to graphical user interface (GUI) based browsers such as Mosaic rose, WWW protocol gained increasing popularity and started to dominate the other protocols in terms of generated traffic.

Due to ever increasing growth of the Internet and the popularity of the WWW protocol, it became increasingly difficult to find resources and information on the Web. Need for directories organizing information, and Web scale search engines, facilitating to find the relevant resources gradually became apparent. In 1994, Galaxy, the first searchable directory was launched. The same year, one of the first search engines, Lycos claimed to index 54K document.

During the mid-90s, an increasing number of Web scale search engines started to appear, offering different alternatives and making meta search feasible. In 1995, Altavista, shortly after its introduction, became an instantly popular search engine due to its features. Metacrawler, one of the first meta search engines, combining and presenting the results of multiple sources, was also launched in 1995. Among various search engines that appeared during the late 90s, Google by Page and Brin (Sep 1998), is the first documented search engine making use of global scale link analysis. Google gained increasing popularity and became one of the leading Web scale search engines. Also in 1998, Directhit made use of click information in ranking the pages by boosting the scores of the URLs that are clicked on by its users. Although simplistic, this approach is one of the first (and few) examples of ranking algorithms based on usage statistics.

In terms of the algorithms that are used, early search engines were not much different than classical information retrieval systems. Score of a page mostly depended on the page alone (except for weighting the importance of the terms). Traditional approaches such as term frequencies and inverse document frequencies in vector-space model were commonly used. Table 1.1 provides an overview of the major algorithms used in Web search engines. Usage of hyperlinks and link analysis for ranking is relatively recent (started around 1998 and accelerated afterwards). Google, one of the

1995	Early search engines use term index and frequency information for retrieval and ranking. Spammers use hidden list of keywords to improve chances of a hit.
1998	PageRank algorithm is introduced by Brin and Page and used in Google.
1999	HITS is used for focused crawling, improving ranking quality of early search engines at the expense of bandwidth and computation.
2000s	Transition to link analysis: Major search engines use link analysis to improve ranking. Spammers now need to modify more than a single page to boost the score of a given page significantly.
2002	Topic sensitive link analysis and ranking.
	What is next? Usage statistics?

Table 1.2. A brief history of ranking approaches used in Web search engines

earliest search engines that uses link analysis has also been launched in 1998.

Before Google, one of the important problems early search engines faced was spamming, which was relatively easy at the time. Spamming refers to purposely altering/boosting the score of a page by some method so that it appears earlier in search results, or so that it appears in a query results even though it ordinarily would not. If link analysis is not used, this can be achieved by embedding a number of hidden keywords that does are not typically visible to the users, but visible to the crawlers. Using this approach, by modifying the page alone, the spammer could increase the chances of getting a hit in popular queries. Similarly, it is also possible to modify the score of a page for a given query by modifying the term frequencies (by adding/modifying terms).

When link analysis algorithms such as PageRank is used, the above, simple approach is no longer sufficient to significantly boost the score of a page. Instead of modifying the page alone, one typically needs to build/enchance the pages/links that affects the score of the page depending on the link analysis algorithm used. Hence, link analysis, PageRank algorithm in particular, rendered the early approaches of spamming mostly absolute. However, usage of link analysis, a new type of information, opens up new doors that can be abused by the spammers. Although it requires relatively more effort, it is now possible to boost the score of a page without altering the page itself, by pointing to it from a large number of pages, or pages that have high scores. Since the latter approach is relatively harder than early approaches in general, link analysis is considered to offer resistance or relative resistance against spamming. Here, and then on, the term “resistance” in the context of an algorithm with respect to spam is used to denote that the algorithm reduces the effects of spam or makes it more difficult. Complete immunity against spam should not be inferred.

Usage augmented link analysis has the potential to offer another layer of resistance against spamming. When some of the algorithms/modifications examined in this thesis (Usage aware PageRank in particular) is carefully applied, in order to significantly boost the score of a page, building/altering the link structure around that page is no longer sufficient. The modified pages/links should also be supported by a sustained traffic received from a large number of sources. The other side of the coin is that, just like the incorporation of link analysis into ranking, a new type of information is used: usage statistics. If necessary precautions are not taken, this information type may introduce additional, perhaps easier ways that can be used for spamming. When carefully applied however, existing spamming approaches may become less effective. In the case of Web scale search engines, whether or not this new signal makes the overall ranking algorithms less prone to spamming remains to be investigated.

1.2 Related Work in Link Analysis and Usage Statistics

Original PageRank [7], developed by Brin and Page, was based on a random walk model and formed a probability distribution over all pages. It was used as the core algorithm in Google, a widely used commercial search engine. The early formulation had a few problems such as “rank sinks” and “dangling links”, and required that the nodes were strongly connected to satisfy convergence properties. It also required that each node had at least one outgoing link to satisfy the probability distribution property, otherwise the PageRanks of all pages would add up to less than 1. The early issues were addressed by a few modifications [37, 20], for instance, by treating each page that has no outgoing link, as if they were pointing to every other page in the set.

Since then, a number of extensions have been proposed. Two of these extensions involve introduction of topic information in order to assign PageRanks that are topic oriented. Richardson et al. [41] precalculated different PageRank vectors for a given number of terms, focusing on the subset of pages that contain the term of interest. For each selected term, a separate PageRank calculation is carried out within the subset of documents that contain the document. When the query contained one or more of these terms, precalculated scores associated with the terms would be used. Haveliwala [19], instead of dividing the set according to topic terms, extracted topic information from Open Directory, and calculated a PageRank vector for each of the categories by boosting the importance of the pages belonging to the category. Jeh et al. [22] addressed the personalization of PageRank scores by segmenting the graph into subgraphs and precalculating scores on the subsets enabling on the fly combination of scores. Kamvar et al. [23] proposed accelerated PageRank computations by inserting intermediate steps, trying to eliminate first two highest non-principal eigenvalues faster. Note that almost all of these extensions, including the modifications introduced in this thesis, use the power method, and have similar high level mathematical formulations. One of the key differences is the way the adjacency matrix is calculated, filtered, or modified. At the end, most of the current link analysis approaches boil down to a similar iterative process.

Link analysis has also been used in other contexts. Another major algorithm for ranking pages by making use of the link structure is the HITS algorithm [25] introduced by Kleinberg. Unlike PageRank, this particular algorithm assigns two scores to each page, an authority score and a hub score. It has a recursive definition that can be summarized as: A good authority is a page that is pointed by good hubs, and a good hub is a page that points to good authorities. The algorithm is primarily used in ranking pages via focused crawls, and in community analysis in a limited way [15]. The original version was not as scalable as PageRank and had convergence problems with high number of nodes. Many extensions have also been proposed for this algorithm. There are also related algorithms. Salsa [29], for instance, is also used in focused crawling, and assigns authority and hub scores to pages. Unlike HITS however, it has a limited random walk model and is not based on a mutual reinforcement model like HITS. A detailed overview of various link analysis approaches in the context of focused crawling can be found in [6].

Fagin et al. compared searching within the Web vs. searching within a large scale, private, corporate network (IBM) [13]. They reported that the IBM network has about 7K hosts containing about 50M distinct URLs, majority of them being dynamic pages. In a number of ways, the IBM Intranet is found similar to the Internet (in terms of page structure and major statistics such as in-degree and out-degree of pages etc.). However some differences are pointed out. The following axioms are proposed as key features in intranets:

- Intranet documents are often created for simple dissemination of information, rather than to attract and hold the attention of any specific group of users.
- A large fraction of queries tend to have a small set of correct answers (often unique), and the unique answer pages do not usually have any special characteristics
- Intranets are essentially spam-free.
- Large portions of intranets are not search-engine-friendly.

A rank aggregation approach is used and two sets of queries are tested. They observed that, for both of the query sets anchortext and PageRank improved ranking quality, while title, content, words in the URL, and discovery date improved ranking quality in one of the sets. Usage information is not used in this study.

A preliminary suggestion for incorporating usage information into link analysis was proposed by Zhu et al. in a short paper [56]. The suggested formula was named PageRate. Although it was claimed to be an extension to PageRank, it did not have the basic PageRank properties. The formulation was not intuitive. Perhaps largely due to lack of experiments, the potential problems in the formulation that could have been identified were not addressed. Normalization was done on incoming links, negating the difference between heavily and rarely used pages, also requiring non-trivial normalization steps. Another suggestion for using usage information was briefly proposed by Miller et al. [33] in the context of HITS, replacing the adjacency matrix by another matrix based on link usage information. No experimental results were presented, except mentioning that the algorithm was applied to a set of 300 pages, and stating that the given approach ranked the main page highest (unlike the other modification, which was the focus of the paper).

Usage information has also been used in web search and related domains in other contexts. Schapira [43] used a reinforcement learning approach for reranking the search results. For each query, the system kept track of how many times a particular URL was clicked on by different users. Similar to Directhit [12], a commercial search engine, when the same query is issued at a later time, past information was used to boost the scores of the frequently visited URLs. Oztekin et al. used positions of user clicks in evaluating various merging and reranking approaches in meta search domain using implicit relevance feedback [36]. Usage information has also been used in profile based systems to learn user interests in time, and to rerank the search results to reflect them (e.g. by changing the relative importance of the terms according to the user's profile learned by the system) [54], [28].

1.3 Chapter Layout

This thesis is organized as follows:

Chapter 1 summarizes the motivation behind investigating the applicability of usage information as an alternative signal for site specific and intranet search. A brief history of the Internet and Web search engines, and selected related work in the area of link analysis and usage statistics are also provided.

An overview of the general ranking procedure used in various search engines is given in Chapter 2, Section 2.1. Leading link analysis algorithms, PageRank and HITS, are described in further detail in Section 2.2. Suggested usage based algorithms augmenting these existing methods, Usage aware PageRank(UPR) and UHITS, along with a purely usage based approach are introduced in Section 2.3. Furthermore, in the case of UPR, a potential improvement, as well as suggestions for implementation are also provided.

Various pieces of the infrastructure, including the site specific search engine built for the purpose of testing the ranking algorithms examined in this study in a real-life search setting, are discussed in Chapter 3. Section 3.1 describes UMirror, the crawler built for USearch. Once a local mirror is obtained, various pre-processing steps that are taken are described in Section 3.2. Section 3.3 summarizes how various scoring vectors are computed for each algorithm. Finally, Section 3.4 describes how these score vectors are incorporated and used within USearch.

Experimental results are provided in Chapter 4, comparing the proposed and existing methods from various angles. Section 4.2 compares PageRank and UPR in detail in terms of global orderings, investigating the effects of usage emphasis on the algorithm. The parameter space for suggested methods are sampled, and all methods are compared against each other in terms of pairwise correlations and distinguishing power they offer (Section 4.3). In the last part of the experiments, selected methods/parameters are compared in a query dependent setting using a modified version of USearch (Section 4.4). All graduate students and some of the faculty were invited to participate in this stage. More than 100 queries were issued and evaluated.

Finally, discussions and concluding remarks are provided in Chapter 5 and Chapter 6 respectively.

Chapter 2

Ranking in Search Engines

2.1 General Ranking Procedure

In majority of recent search engines a number of quality measures or scores are combined to calculate the relevance score of a page for a given query. Classical information retrieval systems typically uses the vector-space model approach, which is also heavily used in recent search engines in conjunction with other quality measures. In general, the similarity score of a document with respect to a query is combined with other quality measures such as PageRank score to obtain the final score of the document for that query.

2.1.1 TF-IDF based Ranking in Vector-Space Model

In vector-space model, each document is converted to a (sparse) vector where each entry is a pair of numbers: id number of a term/word/keyword, and the frequency or weight of that term within the document. Alternative retrieval approaches, and the importance of the terms within a document and the document collection are well studied in the information retrieval literature, early discussions appearing in text books as early as late 70s [49]. The importance of the terms within a document collection can be typically approximated via variations of TF-IDF weighting approach [42] [18]. TF generally refers to term frequency, capturing the frequency of the term/word within the document. IDF refers to inverse document frequency of the term/word, capturing its importance and distinguishing power. It is desirable to assign high IDF scores to terms that have a higher distinguishing power within the collection compared to terms that has low distinguishing power. If a word appears in almost all of the documents within the set, its IDF score would be significantly lower compared to a word that appears only in a small subset of the documents.

IDF score of a term t can be calculated as follows:

$$idf_t = \log\left(\frac{NDoc}{D_t}\right) \tag{2.1}$$

Where $NDoc$ is the total number of documents in the collection, and D_t is the number of documents containing the term t.

Once IDF scores are calculated for all terms, each document vector can be normalized via the IDF vector by multiplying the TF score of each term by its IDF score. TF score of a term t in document d, can simply be the frequency of the term within the document:

$$tf_{t,d} = \frac{N_{t,d}}{N_d} \tag{2.2}$$

or using another common approach:

$$tf_{t,d} = 0.5 + 0.5 \cdot \frac{N_{t,d}}{N_d} \tag{2.3}$$

where $N_{t,d}$ is the number of occurrences of term t within the document d, and N_d is the total number of occurrences of all terms within the document d.

Note that, for some domains or collections, the TF-IDF mechanism is not always sufficient. In addition to TF-IDF, it is also common to use a stop list, in which a number of terms (depending on the language used) can be completely ignored in the context of a query (e.g. terms such as "the", "a", "is".). In other words, depending on the implementation, the stop list may effectively force the IDF scores of the terms within the list to be zero.

The corpus may also be broken down into logical collections, and analyzed independently. For instance, a search engine may try to detect the language or topic of each page and process them within smaller collections such as pages related to health, or pages in a particular language where the frequency of the terms may be significantly different than other collections.

In vector-space model, it is common practice to normalize the vector representing each document, such that its second norm ($L2$ norm) is one, after which the similarity score of a document for a query, can be calculated by taking the dot product of the query vector with the vector representing each document. If the query vector is also normalized similarly, the dot product would result in a score between 0 and 1, where a score of zero shows that none of the terms in the query are present within the document, and a score of one shows that the query vector is the same vector as the document. The above approach is also refereed to as the "cosine similarity" of a document with respect to a query. Note that the dot product of two $L2$ normalized vectors is the cosine of the angle between the two vectors.

2.1.2 Bigrams, Trigrams, n-Grams

A bigram refers to two words that occur together frequently in a given order, forming a logical entity, usually referring to a concept different than its individual components. This analysis can be generalized to trigrams, and n-grams, which are formed with 3, and n consecutive words respectively. It can be further generalized to collocations including preferences in choosing possible terms and multiple forms such as preferred adjectives that goes with a given term.

The term "cell" in "red blood cell" and "cell phone" does not refer to the same concept as the single term "cell". In such cases, weights associated with keywords in the TF/IDF scheme (Section 2.1.1) without using n-grams may not distinguish between the individual concepts represented by the term alone, and various concepts corresponding to possible n-grams formed by the term. It is possible to augment the analysis by considering frequent bigrams, trigrams, and/or n-grams, and indexing them as single terms. For instance, "New York", or "red blood cells" may be treated as single entities or terms within the TF/IDF framework.

While considering possible n-grams within a corpus, the Apriori algorithm, and extracting frequent item sets [1] can be considered . The optimization relies on the fact that in order for an n-gram to occur at least k times, its subsets must also occur at least k times. In this context, the algorithm first considers unigrams, extracting the ones having a frequency (or support) greater than a threshold. On the next iteration, while considering bigrams, possible candidates that have at least the desired support are limited via the possible sets formed by the selected unigrams. At each iteration, items extracted in previous iterations are used to limit the candidate set.

If a pure statistical approach is used, while considering a bigram AB, probability/frequency of A's occurrence; probability/frequency of B's occurrence; and probability/frequnency of both A and B occurring as a bigram, can be used in determining whether or not AB should be considered as a bigram within the collection. A naïve implementation may simple consider the ratio:

$$\frac{P(AB)}{P(A) \times P(B)} \tag{2.4}$$

Assuming that the individual occurrences are fairly independent and the terms do not form a bigram, the probability of having AB as a bigram, $P(AB)$, is expected to be closer to $P(A) \times P(B)$. Hence the ratio would be closer to 1. On the other hand, if the ratio is significantly higher, this may suggest that A and B tend to occur as a bigram AB with higher probability than random coincidence. Note that in natural languages, the independence assumption is a simplification; it does not hold in general (e.g. probability of having the terms "the", and "of" occurring as "the of" may be lower than the probability associated with the individual terms multiplied. Whereas the probability of the terms occurring as "of the" may be significantly higher than the former).

The approaches above can further be improved via natural language processing, for instance, by annotating the terms and restricting the set of possible n-grams to preferred order of nouns, adjectives, verbs etc. An overview of various approaches for collocation and n-grams analysis can be found in [31].

2.1.3 Stemming

Stemming refers to the process of mapping variations of a word to a "stem" of the word. For the English language, Porter's stemming algorithm [39], or its variations are commonly used. For instance, using the above algorithm, words such as "computes", "computing", and "computed" are all mapped to the same stem, "comput". Stemming algorithms for various languages have also been proposed.

Stemming is used in some search engines and information retrieval systems to enable querying variations of a given keyword by providing one variation or the stem of the keyword. It can also reduce the dimensionality of the system by limiting the vocabulary to word stems. This approach has the potential to offer a more manageable vocabulary if resources are scarce. The downside is that, if the user is interested in only one variation of the keyword, results may be cluttered with the other variations of the keyword as well.

Stemming may improve recall in more specific queries by effectively expanding the query with the variations of the keyword. For broader queries on the other hand, if the recall is sufficiently high, stemming can either be omitted or the results may be ordered to give emphasis on the variation supplied by the user within the query text.

The characteristics of a given language may also affect the incentive in using stemming. English is one of the languages in which variations of a given term such as a verb are relatively few. In other languages, it may be common to modify the term by suffixes/prefixes, or the verb conjugation may lead to a larger number of variations. As a result, the degree of stemming, and where and when stemming should be applied may depend on the particular language.

The TF-IDF mechanism and stop list can also be applied with stemming with minor modifications. One simple approach if stemming will be used on all terms, may proceed as follows: Once the stop words are eliminated, the document is supplied to a stemmer, in which all keywords are mapped to their stemmed version before processing further. Similarly, query terms are also stemmed prior to conversion to a sparse vector.

2.1.4 Additional Information Available to Search Engines

Unlike plain text, HTML documents that are typically indexed by search engines have a number of additional information that can be used in ranking. While the TF-IDF approach works fairly well on plain/unstructured text, HTML structure and tags can be used to modify the weights of the keywords further [7], [10], [34], [55]. For instance, terms that appear in the title or headings, or terms that are emphasized or bold-faced may be boosted in weight depending on the importance given to different HTML tags/structures.

Another additional information available to search engines is the use of hyperlinks and the corresponding anchor text. The anchor text is usually a small description of the hyperlink between the anchor tags or a small window (typically up to a few dozens of bytes) around the hyperlink. For instance, Google is reported [7, 16] to augment the text of a document with the anchor texts of the links pointing to the document. Anchor text is also used in focused search in the context of HITS to alleviate the topic drifting problem, by using a stopping criteria based on the similarity of the anchor text to the query [4].

In addition to the above, the hyperlink structure is heavily used in recent search engines via a number of link analysis methods. Some of these approaches typically assign a precalculated global score to each document by a scheme based on iterative weight propagation through the hyperlinks (e.g. PageRank). Query similarity score is then combined with the global score(s) to calculate the final score of a document for a given query. Most of these approaches (including classical PageRank) solely depend on the hyperlink structure and can be precalculated independent of the query. More recently, there are suggestions to assign multiple scores to each document for predefined contexts/topics ([41], [19]). When a new query is issued, the context is determined, and one of these scores are selected and used accordingly. Note that, although current literature does not explicitly address fuzzy selection of topics, there is no reason why more than one score per document cannot be selected and combined in topic sensitive approaches with varying membership degrees. Among recent search engines, Teoma [47], when first launched, claimed that a context sensitive ranking approach was used in the engine. However, additional details were not provided.

Section 2.2 discusses two leading link analysis algorithms in further detail. Proposed usage based methods augmenting these two major algorithms via usage statistics as well as a naïve, purely usage based ranking approach are introduced in Section 2.3. These algorithms, discussed in the remainder of the chapter, produce a single, global score for each document. All of these approaches are treated as a quality measure that is intended to be combined with the query similarity score (along with other quality measures) while calculating the final score of a document for a given query.

2.2 Leading Link Analysis Algorithms

In this section, two major link analysis approaches are discussed. Modifications to these algorithms incorporating usage statistics will be proposed. The modified versions will then be compared to the original algorithm.

2.2.1 PageRank (PR)

PageRank is one of the leading link analysis approaches, proven to be very useful in Web scale search. In conjunction with classical information retrieval approaches, it has been an important quality measure in various search engines, including Google [16].

Original PageRank formulation is based on a random walk model and forms a probability distribution over all pages. The random walk user, either starts from a new page (with probability $1-d$), or follows a random link from the current page (with probability d). The score of a page p is given by

$$PR(p) = \frac{(1-d)}{n} + d \times \sum_{i \to p} \frac{PR(i)}{C(i)} \tag{2.5}$$

where d is the damping factor, n is the total number of pages in the dataset, $PR(i)$ is the score of page i, $C(i)$ is the number of outgoing links on page i, and i iterates over the set of pages that points to page p.

The parameter d controls how much emphasis is given to score propagation (right portion of the formula), vs. initial scores (left portion) during the iterations. If d is low, a page's score is closer to its initial assigned score ($1/n$). Also, since score propagation has a lower weight, convergence is achieved faster. If d is high, score propagation is weighted more, scores of pages that are pointed to by many (important) pages have increasingly higher scores, and convergence is achieved slower in general. The proposed value for d in the original paper is 0.85. Various values for d has been tried in our previous experiments, and 0.85 and similar values turned out to be a reasonable choice to be used in PageRank, as well as the usage based modification that we developed.

PageRank has a number of desirable properties. It has an intuitive interpretation, offers relative resistance against spamming, scales well to the Web, and it is relatively stable (in general, changing a subset of nodes/links does not affect overall scores dramatically). If the graph is fully connected, and if all pages point to at least another page in the dataset, PageRank forms a probability distribution and is inherently L1 normalized: sum of all scores add up to 1. If not all pages have at least one outgoing link, they can be treated as if they point to every other page in the dataset in order to conserve the probability distribution property [37, 20].

2.2.2 HITS

Original HITS algorithm has been proposed by Kleinberg [25]. Unlike PageRank, two scores are assigned to each page: An authority score, and a hub score. The formula can be summarized by a recursive definition: A good authority is a page that is pointed by good hubs, and a good hub is a page that points to good authorities.

For a page p, a nonnegative authority weight a_p and a nonnegative hub weight h_p are assigned. All a and h values are initialized with a uniform constant. The authority and hub weights are updated using the following equations [17]:

$$a_p = \sum_{i \to p} h_i \tag{2.6}$$

$$h_p = \sum_{p \to i} a_i \tag{2.7}$$

Similarly, in matrix form, if A is the $n \times n$ adjacency matrix ($A(i, j)$ is 1 if i points to page j, or 0 otherwise), and a and h are the authority and hub vectors, the two score vectors can be iteratively calculated:

$$h_{i+1} = A.a_i \tag{2.8}$$

$$a_{i+1} = A'.h_i \tag{2.9}$$

where, A' is the transpose of A. Note that, after each iteration, the two vectors are L1 normalized, since, unlike PageRank, the formulation does not have an inherit auto-normalization property.

HITS is primarily used in the context of focused crawling. A query is issued to a search engine and top n documents are retrieved to form the core document set. The set is then expanded by following both incoming and outgoing links to/from these pages (typically, going 2 links forward/back, resulting in a few thousands of documents). Usually topic drifting is handled by limiting the expansion when query and content/anchortext similarity drops below a threshold. Once the document set is formed, hubs and authorities are calculated using HITS. HITS can also be applied globally. However, for large number of documents, it has scalability problems, and is not as stable as PageRank in general.

	Uses static link structure	Uses link traversal statistics	Uses page visit statistics
Naïve	no	no	yes
UHITS	yes (optional)	yes	no
UPR	yes (optional)	yes	yes

Table 2.1. Key differences between proposed usage based ranking methods

2.3 Proposed Usage Based Ranking Methods

A selection of usage based ranking methods are proposed to cover a wide range of usage types. Usage aware PageRank (UPR) uses both link traversal statistics and page visit statistics, HITS modified via usage (UHITS) uses link traversal statistics, and finally two versions of the naïve approach Counts and MCounts use page visit statistics. Both UPR and UHITS augment the corresponding link analysis approach with usage information. How much emphasis is to be given to usage vs. static link structure is controlled via one or two parameters. Counts and MCounts, on the other hand, are not based on link analysis. They are primarily based on the number of times a given page has been visited.

High-level differences between implemented usage based ranking methods are summarized in Table 2.1. These methods are discussed in further detail in the following sub-sections.

2.3.1 Usage Aware PageRank (UPR)

The random walk user in the original PageRank formulation does not differentiate between links on a given page while deciding which one to follow, nor does it differentiate between pages when it decides to start from a new page. Consider a scenario where we have a page, p_1, which is bookmarked by various users, therefore visited more often than other pages. Consider a link on that page, $p_1 \rightarrow p_2$, that points to page p_2, which is followed by majority of its visitors, but other links on page p_1 are hardly used. Intuitively, the probability of users visiting page p_1 is higher than the probability of users visiting other pages. Similarly, PageRank of p_1 should contribute more to the PageRank of p_2, since $p_1 \rightarrow p_2$ is followed more often than other links on p_1. The random walk model does not capture these differences. We modify the user model in favor of a biased random walk model. The biased random walk surfer differentiates between links as well as pages. It does not follow each link with equal probability. It also has preferences while selecting a new page to visit without following a link. In our model, score of a page is distributed to the pointed pages, proportionally to the probability that the average user will follow the corresponding link. Also, the probability distribution associated with starting from a new page is no longer uniform, but biased by the estimated probabilities. Using

UPR, score of a page p is given by

$$UPR(p) = (1-d) \times \left(\frac{1-a_1}{n} + a_1 \times W_{jump}(p)\right) + d \times \left(\begin{array}{l} (1-a_2) \times \sum_{i \to p} \frac{UPR(i)}{C(i)} + \\ \qquad a_2 \times \sum_{i \to p} \frac{UPR(i) \times W_{link}(i \to p)}{W_{total}(i)} \end{array} \right) \tag{2.10}$$

where the parameter a_1 controls the usage emphasis in the random jump portion of the formula, a_2 controls how much usage emphasis will be given in calculating the weights of the links, $W_{jump}(p)$ is the estimated probability to go to page p directly without following a link, $W_{link}(i \to p)$ is the weight of the link from page i to page p in the usage graph, and $W_{total}(i)$ is the total weight of all outgoing links from page i in the usage graph. Note that $W_{link}(i \to p)/W_{total}(i)$ estimates the relative probability that the average user will follow the link $i \to p$ from page i to page p (as opposed to $1/C(i)$ in the original formulation).

The parameters a_1 and a_2 act as sliders and control how much emphasis is given to the usage information vs. the static link structure. If they are set to zero, the formulation becomes equivalent to the original PageRank formulation; if they are set to one, the emphasis completely shifts to the usage graph; for values in between, the behavior gradually shifts from PageRank to pure usage based estimations.

The motivation behind using two parameters as opposed to a single one may not be apparent at first glance. However, the two portions of the formula have different characteristics, and one may choose to adjust them separately. In some sense, the first portion of the formula states how the scores (or the flow) will be injected into the system through each page. In the case of PageRank, the influx from each page is fixed and proportional to $1/n$, while in the case of pure usage based approach, the influx from each page is proportional to the probability that the average user will jump to that page directly. The second portion of the formula states how the injected scores will flow through the links. In the case of PageRank, scores are equally distributed across links, while in the case of pure usage based approach, scores are distributed according to the probabilities of following the links.

Deemphasizing the usage statistics associated with visiting a page directly may be desirable in some cases, for instance, when we want to reduce the effects of browser home pages (which may get unintentional hits every time a user opens up the browser), and when we do not have full usage statistics for a subset of pages. Note that weights of links going out of a given page is normalized for each page in our formulation. If we do not have usage statistics about the links of a given page, all of them can be treated equally. However, if we assign initial weights reflecting the number of hits on a page without following a link, portions of the dataset that do not contain usage statistics will be penalized. Note that a similar situation occurs if the usage sampling rate for different portions, domains, or sites are different; the portions that are undersampled will be assigned low initial scores. By using two independent parameters, it is possible to reduce such undesirable effects in the presence of partial or uneven information, without affecting the second portion of the formula. Note also that, custom browsers or crawlers can choose to omit the referrer field altogether. It is also relatively easier to attack the first portion of the formula via usage based spamming if the first parameter is set to a high value, close to one. If a significant subset of browsers/tools does not use the referrer field, or if usage based spamming is suspected, the first parameter can be lowered without affecting the second portion and the global stability of the formula.

Implementation Issues

A simple way to approximate the probabilities is to use the frequencies or by simply counting the number of hits to pages/links. In this approach, $W_{jump}(p)$ is the number of visits to page p when the referrer field is empty divided by total number of visits to all pages with an empty referrer field; $W_{link}(i \rightarrow p)$ is the number of times the page p is accessed when the referrer field is page i; $W_{total}(i)$ is the total number of times a link is followed from page i (referrer field is page i). This approach is tested first and worked reasonably well for most cases. However, there are certain types of user behavior that may bias the probability estimates. Every time a user opens up a browser, the browser's home page gets a hit, even if the user's intent is to visit another page. Also, one may argue that equal number of hits to a page/link from several users should be weighted higher than a user hitting the same page/link several times, giving precedence to the overall group behavior as opposed to individuals. In the UPR formulation, instead of using counts of links followed, and counts of hits to pages with empty referrer fields, we can deemphasize successive accesses to pages/links from the same user in a given time window by using modified counts which is a log transform of the counts:

$$MCount = \log_2(1 + Count) \tag{2.11}$$

If there are no accesses to a particular page, the modified count is $\log_2(1 + 0) = 0$, if there is a single access, the modified count becomes $\log_2(1 + 1) = 1$, and the modified count lowers the weight of subsequent accesses from the same user. After the transformation, adding up all the modified counts from all users for all time windows, gives us estimates that better reflect overall user behavior. This approach has the additional benefit of making the system more robust against usage based spamming. Note that, if n people access a link/page just once within the time window, the modified counts will be exactly the same as before. However, if only a single individual accesses the same link/page n times the contribution of this time window will be $\log_2(n)$.

The first approach (simple counts), does not require time and IP information. However, in order to implement the modified counts approach, we need a way to identify users or user groups. IP numbers or anonymized IP numbers/user identifiers can be used for this purpose. We also need to divide the dataset into different time windows. For this purpose, time stamps can be used. In our site, we found a day to be a reasonable time window, which is consistent with the way the logs are rotated. It is also fairly easy to divide the aggregated logs into time windows of a day.

UPR is similar to PageRank in terms of computation and is also quite scalable. PageRank can be implemented in terms of sparse matrix-vector operations, at each iteration, matrix vector product being the most costly operation (weighted adjacency matrix and PageRank vector). Once the adjacency matrix is modified for UPR, subsequent operations are similar. Only the differences will be illustrated: The full usage graph is stored in sparse format, and requires only slightly more space than storing the static linkage graph (for each link and page, we need a floating point entry to store the weight capturing the frequency/probability information). Usage information can be built/updated incrementally. The simple counts approach is trivial and will be omitted. Modified counts approach is implemented efficiently as follows: Usage graph is incrementally built or updated by processing each time window separately. The full graph is stored on secondary storage (in sparse format). When new data is available (a new time window is built), only information about the links/pages that appear in the current time window needs to be efficiently accessed and updated. In memory hash tables or balanced trees can be used to insert and update new links/pages. Current time window is scanned in one pass, and a partial graph and statistics corresponding to the entries in the log is built in memory. Once the entries in the current time window is incorporated, the partial graph is merged with the full usage graph in one pass on the secondary storage. For instance, if balanced trees are used, the space complexity for processing a time window is proportional to the number of unique pages/links that appear in the time window. If there are n unique pages/links in the time window, the time complexity for processing the full time window becomes $n \times log(n)$ using balanced trees. Other data structures are also feasible, some offering different storage/time tradeoffs. It is also fairly easy to decay old usage information, for instance, by using an exponential decay approach while merging the full usage graph with the partial graph corresponding to the current time window.

Once the usage graph is updated, a lazy UPR iteration does not take more than a constant times higher than a PageRank iteration. The space complexity is also no more than a constant times higher. Once a_1 and a_2 are fixed, it is possible to rearrange the formula and build a combined matrix to be used in further iterations. After this precalculation step, UPR iterations become similar to PageRank iterations in terms of speed and storage. The rearranged formula is as follows:

$$\begin{aligned} UPR(p) = (1-d) \times \left(\frac{1-a_1}{n} + a_1 \times W_{jump}(p) \right) + \\ d \times \sum_{i \to p} UPR(i) \left(\frac{1-a_2}{C(i)} + \frac{a_2 \times W_{link}(i \to p)}{W_{total}(i)} \right) \end{aligned} \tag{2.12}$$

Note that the only term that changes during the iterations is $UPR(i)$.

UPR inherits basic PageRank properties. Usage information can be updated incrementally, and the formulation is quite scalable. UPR iterations have similar time and space complexity as PageRank iterations (only a small constant times more expensive). Usage importance can be controlled smoothly via the two parameters. UPR can also work with limited usage information: if the two parameters are set to any value between 0 and 1 (exclusive), it gradually converges to PageRank as less and less usage information is available. At the extreme case (complete lack of usage information), it will be equivalent to PageRank. Also, the modified counts approach, increases UPR's robustness and resistance against usage based spamming. Note that the formulations suggested by the two preliminary attempts [56], [33] in incorporating usage statistics into link analysis did not have majority of the key UPR properties (in particular, ability to work with limited usage gracefully, ability to control usage emphasis, scalability, and resistance against usage based spamming).

2.3.2 HITS Modified via Usage Statistics (UHITS)

Miller et al. proposed a modification to HITS in which the adjacency matrix is replaced by another matrix based on link usage information. All entries in the adjacency matrix are initially set to zero. For each link traversal observed in the web logs, the corresponding entry in the adjacency matrix is incremented. Note that if a link has never been used in the web logs, the static link structure is ignored (the corresponding entry is zero).

We generalize this approach and add a parameter a acting as a slider between static link structure and usage based linkage. The composite adjacency matrix is calculated as follows:

$$A = (1-a) \times A_{static} + a \times A_{usage} \tag{2.13}$$

Similar to UPR, if the parameter is set close to zero, static link information is emphasized and the formulation converges towards original HITS. If the parameter is set to one, it should be very similar to the formula suggested by Miller et al. Note that Miller's formulation is not normalized. L1 normalization is used in the adjacency matrix in our version, which makes the formulation more robust against partial lack of usage information or uneven sampling. Unlike UPR, Miller's suggestion as well as the generalized version does only use statistics on link traversals. It does not make use of statistics on going to pages directly (using bookmarks, or typing the URL).

2.3.3 The Naïve Approaches: Counts and MCounts

The two simple quality measures implemented are primarily based on number of visits to pages. Using the Counts method, we basically count the number of times users have visited a given page, and use it as a quality measure. Once counts for all pages are obtained, the scores are L1 normalized. Using this approach, score of a page p (after L1 normalization) is given by:

$$Score(p) = \frac{Count_{visit}(p)}{\sum_{i=1}^{n} Count_{visit}(i)} \tag{2.14}$$

where, n is the number of pages in the dataset, and $Count_{visit}(i)$ is the number of times users visited the page i, either by following a link, or by going to the page directly (typing the URL or using a bookmark).

In the MCounts approach. instead of using direct counts, we use a log transform of the counts similar to Equation 2.11. Score of a page p becomes:

$$Score(p) = \frac{MCount_{visit}(p)}{\sum_{i=1}^{n} MCount_{visit}(i)} \tag{2.15}$$

where, $MCount_{visit}(i)$ is the log transformed counts. Similar to the UPR case, the aim is to deemphasize the scores of pages visited by a few users a large number of times, in favor of pages that are visited by many people. This is achieved in the same manner as UPR using modified counts: The dataset are divided in time windows. For each time window, number of visits from each user are counted separately. The log transform of the counts are obtained, and modified counts for each user are added up for all time windows.

Note that HITS modified via usage solely uses statistics on link traversals. Counts and MCounts methods ignore the link traversals and use number of visits to pages. UPR on the other hand, makes use of both link traversal statistics as well as statistics on going to pages directly.

2.4 Summary

Modern search engines employ various techniques such as TF-IDF weighting, stemming, n-grams analysis, usage of html tags and anchor texts etc. to improve ranking quality. They combine various quality measures and signals to produce an ordered set of results for a given query. For Web search, one of these important signals is link analysis.

Two major link analysis approaches examined in this thesis are PageRank and HITS. PageRank and its variants have proven to be a strong quality signal in Web search and has been used in leading search engines such as Google. HITS and its variants have been primarily used in focused search where a small corpus related to a query or a topic is being retrieved and reranked. Especially PageRank, and in some degree HITS variants, are also used to rank pages independent of a query, offering a topic/query independent global ordering.

Usage information has been previously used in a limited way in ranking. We introduce modified, generalized versions of PageRank and HITS making use of usage statistics. A naïve approach, purely based on number of visits to a page is also examined. Usage Aware PageRank makes use of link traversal statistics as well as page visit statistics. UHITS modifies the adjacency matrix using link traversal statistics only. The naïve approach is purely based on page visit statistics, ignoring linkage information.

These algorithms can be combined with classical information retrieval approaches to produce a final ranking. Each algorithm assigns a global quality score to each document in the repository, producing a quality vector that can be plugged into the USearch infrastructure. Apart from this quality vector, the infrastructure follows the same steps for each algorithm to produce a ranked result set for a given query. All things being equal, this thesis compares the ranking quality offered by these algorithms, examining the impact of usage statistics in an intranet/site specific search setting. The USearch infrastructure is introduced in Chapter 3. Experiments comparing the ranking algorithms are provided in Chapter 4.

Chapter 3

Infrastructure

In order to compare PageRank and HITS, as well as the usage based ranking approaches in a real life search setting, USearch, a flexible site specific search engine has been built. This chapter discusses various pieces of this infrastructure in further detail. Major steps/modules used by the infrastructure can be summarized as below:

- Crawler/site mirroring tool.
- Modules processing the local copy of the site including link extraction, filtering, indexing, and building the site graph.
- Modules processing the usage logs, collecting link traversal and page visit statistics, building the usage based site graph.
- Various modules that calculate the score vectors for documents in the repository (one vector for each algorithm/parameter selection under comparison).
- A general purpose, flexible search engine that incorporates the ranking methods, testing them in a real life setting.
- Modules for analyzing the results including global comparison of each ranking method (via statistics such as Pearson and Spearman Correlations). Exploring the effects of different parameters and usage emphasis values.

The interface for each module is carefully designed, making it possible to replace a given module by another one having a similar functionality when needed. The whole process is broken into different pieces working as a pipeline. Mirroring tool produces a local copy of a given site in a specified format. The link extractor/graph builder module traverses the directory structure; builds a keyword based index using a third party tool; assigns a unique id to each URL; builds the site graph (adjacency matrix); and produces a number of data files and statistics that will be used in further stages. A separate module is called to produce a snippet for each document. Another module processes the usage logs, building a usage based graph in a format similar to the static linkage graph. Then, various modules are called to calculate the scores of internal URLs using various ranking methods and parameters, producing a score vector for each method/parameter combination. After this stage, score vectors and snippets are ready to be plugged into the search engine. The whole process is automated using a number of scripts taking into account various dependencies. Once a new mirror is obtained, naturally, most of the pipeline needs to be recomputed. However, if an intermediate step, such as a given ranking module is updated, only the dependencies that are affected are computed for efficiency and ease of experimentation. High level interaction between various modules are summarized in Figure 3.1.

3.1 Site Mirroring Tool (UMirror)

Various open source/free site mirroring utilities are examined in detail for suitability. The desired criteria in a suitable mirroring tool include:

- Ability to focus on a given domain/site/subsite by regular expression or wildcard based matching (e.g. *.cs.umn.edu/).
- Ability to select the type of items that should be mirrored (HTML files, text files, dynamic HTML files etc.)
- A unique mapping from a given URL to its local copy, and from the local copy to the URL, which may not be trivial when all possible port and protocol variations are considered. [1]
- Ability to identify and distinguish default HTML index name (index.html or index.htm for various sites). [2]

[1]Many mirroring tools does not include protocol information or port information. Combined with poor default html name handling, this can result in potentially serious name collisions in the local copy of a site.

[2]When a link such as http://www.cs.umn.edu/ is to be retrieved, the web server automatically uses the default HTML name. Most crawlers assume that this name is “index.html” or “index.htm”. However, the naming can be quite arbitrary (e.g. for different languages). If the site uses “default.html” as the default HTML name. http://some.site/ and http://some.site/default.html retrieves the same page. In this case, if the crawler assumes that the default html name is “index.html”, and if there is also index.html within the same directory as “default.html”, a name collision occurs. This problem may be especially important when both index.htm and index.html are heavily used within a site.

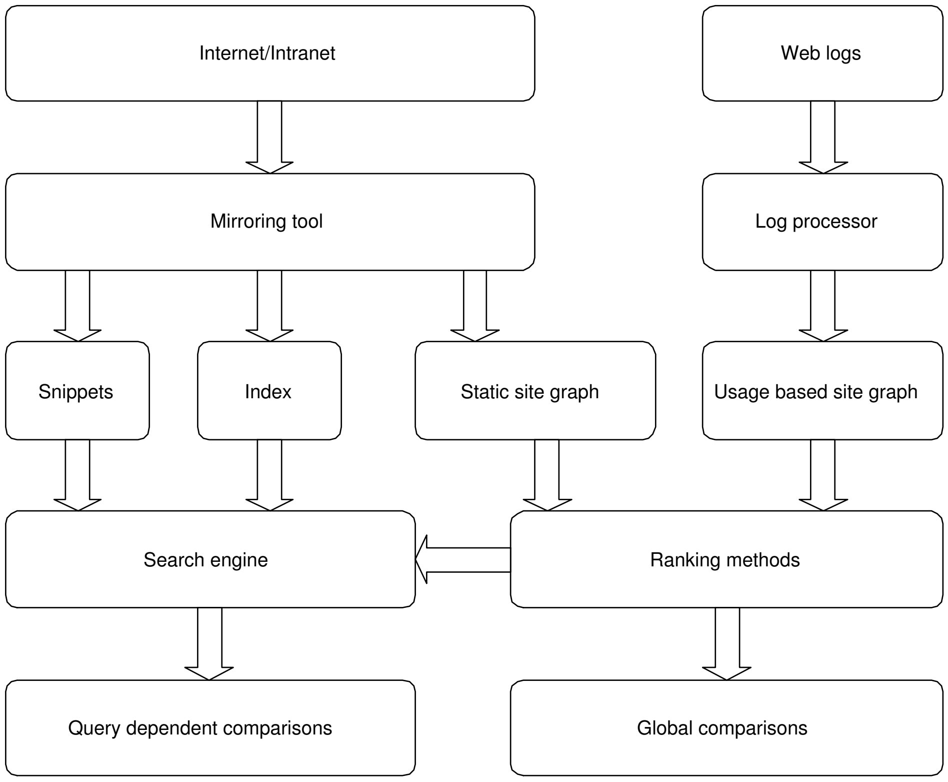


Figure 3.1. Interaction between modules

- Scalability issues (limit on number of pages/items that can be downloaded).
- Support for robots.txt (a protocol specifying which parts of a site can be mirrored and at which rates).
- A reliable queueing implementation to support recursive and stable crawling.
- Support for various protocols other than HTTP. In addition to regular HTTP, FTP, secure HTTP, and secure FTP would be useful for future extensions (optional).
- Support for HTTP 1.1 for efficient resource usage (optional).
- Support for multithreading (optional).
- Portability, support for multiple platforms.

None of the mirroring tools examined covered a large portion of the essential properties. The ones that are worth mentioning include Wget [51], Pavuk [38], HTTrack [21], and cURL [9].

Wget has a number of desired features. It is ported to many platforms and is multithreaded retrieval capable on most of them. It supports different protocols, and robots.txt specification. It has the capability for recursive crawling. It can limit crawling within a site/domain. It is also possible to specify which file extensions are to be crawled. However, it does not produce a unique naming. For instance, https://URL and http://URL are mapped to the same local name, resulting in a name collusion if the same resource exists in different protocols. It also lack a way to specify which default HTML name should be used (assumes index.html).

Pavuk does not support multiple protocols, and is similar to Wget in terms of capabilities. However, it has a better local naming scheme, resulting in a one to one mapping from URL to local name and vice versa. However, both Wget and Pavuk have stability problems. In the test site (cs.umn.edu domain), both of the tools terminated prematurely after downloading about 3K documents.

HTTrack is multithreaded, has name collision problems, and at the time of comparison was inferior to Wget in terms of capabilities.

cURL does not support recursive crawling, but supports various protocols (similar to Wget), and is quite portable. It can potentially be used as a retrieval tool hiding the protocol level details within a larger framework.

As none of the tools that are examined had the essential features that are required, a queueing and link filtering system (UMirror) making use of regular expression matching and HTML parsing is implemented. Wget is used for retrieval as an external call (cURL can also be used with minimal changes). The queueing system is robust and efficient. HTML 4.01 [40] standard is closely followed. The parser checks for validity of the HTML and understands various types of links (not only limited to a href) as well as various ways of converting relative links to global URLs. Although, it is not currently multithreaded, it is very stable, and has a unique mapping from URLs to local directory structure and vice versa. Since Wget is used, the mirroring utility is also highly portable and can potentially support other protocols with little modification.

For default html name, an uncommon name, also used by Pavuk, is selected to reduce the possibility of name collisions (__.__.html). This name can be changed easily by recompiling UMirror. When one of the characters (e.g. '_') is treated as a special character and escaped when fetched directly, but not escaped when it is used as the default html name, name collisions are eliminated completely.

The directory structure that is adopted is as follows: /protocol/domain_port/relativepath/filename For instance, http://www–users.cs.umn.edu/∼oztekin/pmvis/index.html is mapped to /http/www–users.cs.umn.edu_80/∼oztekin/pmvis/index.html. Note that 80 is the default port number for the HTTP protocol and is assumed if it is not overridden. If a special character is used within the URL which would be impractical or unsafe to be used within file/directory names, the character is escaped using URI conventions [3](e.g. %20 is the space character).

Relative links containing malformed "./", "../", and "/" as well as various exceptions such as rarely used HTML features to override the base path are correctly handled. The mirroring tool can be easily configured for a new site via a small configuration file (specifying domain boundaries), and mirror that site in a robust and stable manner, producing a unique mapping from URLs to local directory structure.

As a side note, UMirror does strict HTML 4.01 compliance checks and can detect various anomalies that most browsers ignore. While crawling cs.umn.edu domain using UMirror, various errors within the main site structure as well as in the official HTML pages are detected and reported to the system staff. Problems in the scripts and templates used in generating these HTMLs are checked and fixed to produce a healthier site as a result.

3.2 Processing the Local Directory Structure

Once a site is mirrored, links need to be filtered and extracted to build a static link structure graph. For this, among other modules, HTML parsing and name mapping modules developed for UMirror are used. Link extractor module recursively processes each HTML and extracts links of the specified types, converting relative links to global URLs when needed. Each URL is given a unique id number, which is, in turn, used within the adjacency matrix that is being constructed (for space/computation efficiency). Since most of the pages tend to link to its closer neighbors with higher probability than random pages (e.g. navigational links), simple heuristics can be used to reorder the URLs, resulting in slightly better performance due to better cache hit/miss ratio during matrix/vector operations. The current version orders the pages in a breadth first order. Other trivial reordering approaches such as alphabetical order can also be considered, in most cases, resulting in a noticeable, but slight speedup compared to a random ordering.

The entries within the adjacency matrix are weighted (between 0 and 1). A sparse matrix structure is used. The link extractor can be extended to try different link filtering approaches (weighting links according to path distance, token distance etc., as well as removing some navigation links). Since the links will be weighted according to usage, this feature is currently disabled.

A number of third party programs/libraries are examined for suitability for indexing. Among various choices, Swish-E [44] covered a large portion of the desired properties and selected to be used in indexing the HTMLs. It is portable, light-weight, scalable, efficient, and has support for boolean queries and optional stop word list. HTMLs are converted to text via HTML to text conversion tools developed for Mearf [36, 32], and then indexed.

For each internal URL, the HTML is processed to extract the snippet. 200 bytes are allocated per snippet. All snippets are concatenated in a single, binary file. Once the number(id) of a particular internal URL is known, its snippet can be efficiently fetched from the file by calculating the corresponding offset. During the search process, this procedure requires a single seek and fetch for each subsequent snippet to be displayed once the snippet file is opened, and is implemented efficiently. Note that, for small to medium size sites, the whole snippet file can be cached in the main memory, further speeding the process. It is also possible to distribute the data into a number of machines in a simple manner if scalability issues arise for larger sites.

3.3 Calculating the Scores

Once the static link structure graph, usage graph, and supporting information are available, various modules are called to calculate the scores of internal URLs to build a ranking vector for each method/parameter selection. Matlab's ASCII sparse vector/matrix format is used for compatibility and ease of input/output between C++ and Matlab modules. Once the relevant information is supplied to each module, scores of all nodes are obtained in a sparse format using the specified parameters. Scores of internal nodes are then extracted and collected in a binary file. 4 Bytes are reserved for the score of each internal URL. As in the case for the snippet file, once the URL id is known, the relevant score is fetched by calculating the corresponding offset of the file. Again, for small to medium size sites, this information can fit in the memory easily.

Most of the above steps are carried out via a number of C++ programs, a few Matlab scripts, and a number of scripts automating the generation of various score vectors, calling the modules with pre-selected parameter values for automation.

3.4 Search Engine (USearch)

In order to try various ranking approaches, a general purpose, flexible search engine has been developed: USearch [48]. The current prototype can be accessed at http://usearch.cs.umn.edu/.

USearch engine produces a dynamic HTML file (using CGI) that closely conforms to HTML 4.01 [40] specification. User interface is well-designed, making use of fairly recent HTML features such as Cascading Style Sheets (CSS) [50] to decouple user interface design and choices from the main functionality, while offering the ability to customize a large portion of the output to user needs independent of the main engine. This approach allows a rich user interface for browsers that support CSS. The experience is further enhanced by limited, yet non-vital use of Javascript.

USearch is tested on various platforms using various browsers. Browser dependent features are avoided in favor of commonly available features within the HTML 4.01 specification. Note that the advanced user interface features as well as the use of limited scripting (Javascript) does not necessarily reduce USearch's target audience. If a browser/platform does not support the latest HTML specification and the above features, USearch will still run with full functionality, as long as the browser conforms to HTML 4.01 or previous specifications. In practice, older versions of popular browsers, and new, limited browsers such as the ones on portable devices, PDAs etc. should be able to use USearch with reasonable functionality.

Total HTML size retrieved for each query was also one of the concerns during the design process. Careful use of CSS and decoupling user interface from the main HTML file has the additional benefit of reducing the HTML size. CSS is retrieved once and cached by most browsers. Moreover, formatting related to each item/cell in the result screen is minimal and shared by all cells. By using CSS in formatting and eliminating repetitions, it was possible to reduce the HTML size by more than a factor of two compared to older approaches using tables and naïve formatting.

The engine itself is highly portable. It is written in C++ and has been tested on various flavors of Unix including GNU/Linux and OpenBSD. It also runs on Windows using various Web servers. The engine is quite optimized, efficient, stable, and scalable. Using the test site (cs.umn.edu), a query returning 50 documents out of a database of 20K documents, takes about 0.01 seconds on a

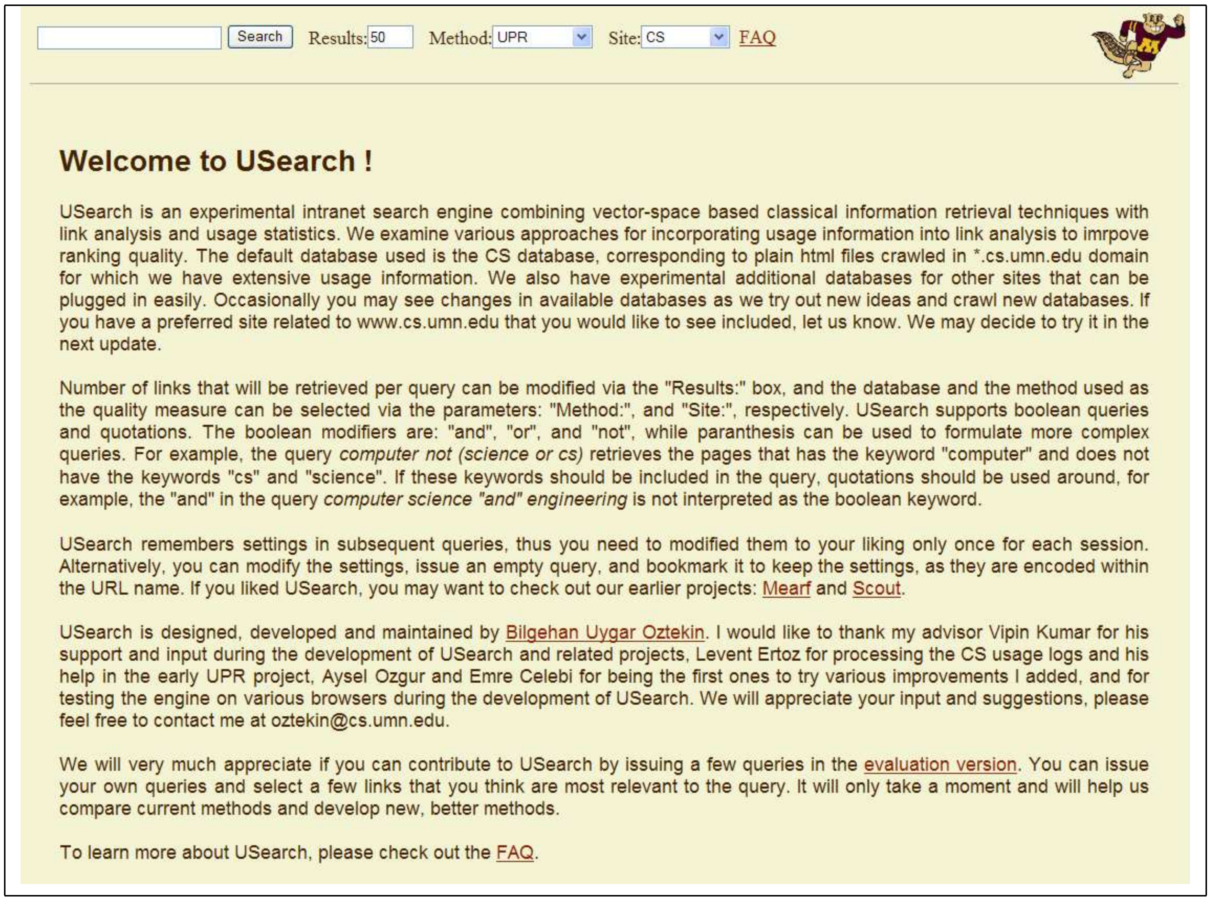
Celeron 333Mhz, OpenBSD system using any of the ranking methods implemented.

USearch is implemented in a modular and extensible way. It is possible to add new databases and new methods with different parameters with minimal effort. It directly uses the index file, and the binary snippet and score formats produced by previous stages in the pipeline. USearch uses a reranking approach. Once a query is issued, top n documents are retrieved (sorted according to cosine similarity to the query). These results are then reranked by combining the text similarity score and the selected ranking score (Final score for a given document is obtained by multiplying the two scores). Note that this approach offers better flexibility while adding, removing, or modifying ranking methods. For a production system, once a final method and its parameters are selected, it is possible to combine the indexing structure with one or a limited number of ranking vectors, thus, eliminating the score combination stage. However, for USearch, where multiple ranking approaches are implemented and compared, this slight gain in speed is not worth the loss of flexibility, considering that the engine is already quite optimized and fast.

Different ranking methods/quality measures can be compared in a query dependent setting in a fairly unbiased way using USearch. All steps (except the particular quality measure under comparison) are the same for all methods. If a method reranks the documents in a way that the high quality documents appear in earlier positions compared to other methods, this method is intuitively better than the other methods for this particular query. By issuing a number of queries, and examining the positions of the "relevant", high quality documents using each method, it is possible to compare these methods for the queries that are examined.

A screenshot of the USearch main page is shown in Figure 3.2. Number of links to be retrieved per query is specified via the "Results" box. The default behavior is to return 50 links per query, and the current maximum value is 1000 links per query. Method selection is carried out using the "Method" box. Available methods are:

- Normal: Use similarity to the query only (no reranking).
- PR: Regular PageRank implementation.
- UPR: Usage Aware PageRank, with a_1 and a_2 set to 0.75.
- Counts: The naïve method using number of visits to pages as a quality measure.
- MCounts: Similar to Counts, but uses modified counting scheme (log transfer of counts).
- HITS-aut: Regular HITS, authority vector.
- HITS-hub: Regular HITS, hub vector.



Welcome to USearch !

USearch is an experimental intranet search engine combining vector-space based classical information retrieval techniques with link analysis and usage statistics. We examine various approaches for incorporating usage information into link analysis to imrpove ranking quality. The default database used is the CS database, corresponding to plain html files crawled in *.cs.umn.edu domain for which we have extensive usage information. We also have experimental additional databases for other sites that can be plugged in easily. Occasionally you may see changes in available databases as we try out new ideas and crawl new databases. If you have a preferred site related to www.cs.umn.edu that you would like to see included, let us know. We may decide to try it in the next update.

Number of links that will be retrieved per query can be modified via the "Results:" box, and the database and the method used as the quality measure can be selected via the parameters: "Method:", and "Site:", respectively. USearch supports boolean queries and quotations. The boolean modifiers are: "and", "or", and "not", while paranthesis can be used to formulate more complex queries. For example, the query *computer not (science or cs)* retrieves the pages that has the keyword "computer" and does not have the keywords "cs" and "science". If these keywords should be included in the query, quotations should be used around, for example, the "and" in the query *computer science "and" engineering* is not interpreted as the boolean keyword.

USearch remembers settings in subsequent queries, thus you need to modified them to your liking only once for each session. Alternatively, you can modify the settings, issue an empty query, and bookmark it to keep the settings, as they are encoded within the URL name. If you liked USearch, you may want to check out our earlier projects: Mearf and Scout.

USearch is designed, developed and maintained by Bilgehan Uygar Oztekin. I would like to thank my advisor Vipin Kumar for his support and input during the development of USearch and related projects, Levent Ertoz for processing the CS usage logs and his help in the early UPR project, Aysel Ozgur and Emre Celebi for being the first ones to try various improvements I added, and for testing the engine on various browsers during the development of USearch. We will appreciate your input and suggestions, please feel free to contact me at oztekin@cs.umn.edu.

We will very much appreciate if you can contribute to USearch by issuing a few queries in the evaluation version. You can issue your own queries and select a few links that you think are most relevant to the query. It will only take a moment and will help us compare current methods and develop new, better methods.

To learn more about USearch, please check out the FAQ.

Figure 3.2. USearch main page

- UHITS-aut: HITS modified via usage, authority vector (a=0.75).
- UHITS-hub: HITS modified via usage, hub vector (a=0.75).

Note that not all method/parameter pairs that are examined are included in the public user interface. A representative from each method with reasonable parameters are selected to reduce clutter.

The database selection is carried out using the “Site” box. Adding a new database (site or domain) to USearch requires the following steps:

- Selecting a unique identifier for the site (e.g. CS for cs.umn.edu).
- Preparing a small script specifying domain/site boundaries and running UMirror.
- Once the mirror is obtained, calling a script that builds the full structure for the specified site.
- If usage logs are available, processing the usage logs and producing a usage based graph.
- Once the score and snippet files are prepared for the site, calling another script that copies the files to the search engine’s input directory.
- Rerunning the search engine after adding the identifier of the new site to the configuration header.

3.5 Summary

USearch is a flexible platform to test various ranking methods. It has fully functional crawling, filtering, indexing, pre/post processing, link analysis, usage log analysis modules, as well as a fairly optimized and flexible search engine that serves the results. It can crawl, index, process, and serve queries for a small to medium size site out of the box with modest hardware requirements under reasonable load.

Almost all modules of the USearch infrastructure (except platform dependent scripts) are highly portable. The engine itself runs on various platforms, and can be used via various browsers/platforms. The user interface is designed in a way that the functionality is still retained if a browser/platform does not support a subset of the advanced features used to enhance the presentation (e.g. older browsers, handheld/embedded devices etc.).

By design, USearch uses a reranking approach, in which top n results in terms of cosine similarity to the query are returned and reranked by the selected method under comparison. Since all steps except the quality measure are the same for all methods, it offers a fairly unbiased way of comparing different quality measures.

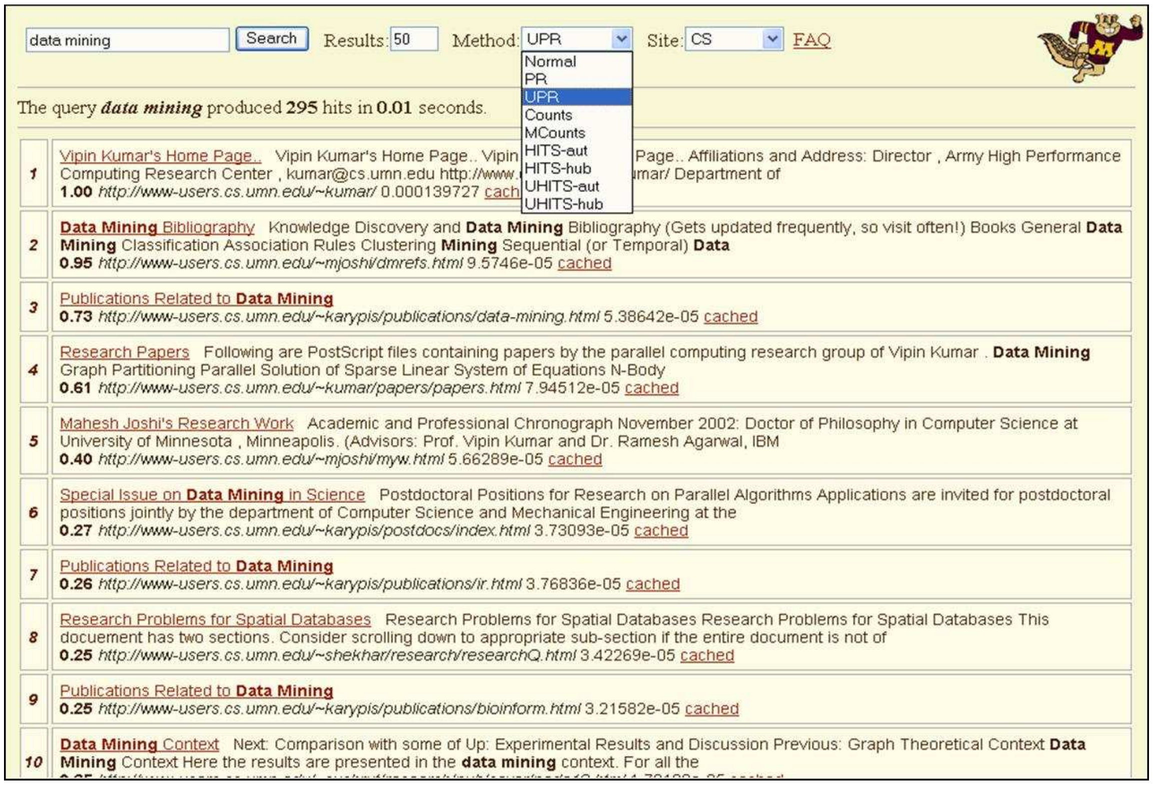
Different methods can be compared in a query dependent setting by issuing a number of test queries and comparing the position of the documents for different methods for the same query. A quality measure that places the important documents in earlier positions compared to another method is deemed superior for that particular query.

Chapter 4

Experimental Results

In order to evaluate the performance of the proposed methods, a number of quality measures are implemented and incorporated within USearch, http://usearch.cs.umn.edu/. Figure 4.1 shows the results of a typical query using the engine. User can select the number of links to be retrieved (default is 50), the quality measure used (PageRank, UPR etc.), and the database to search from (currently, only cs.umn.edu domain has usage statistics available). The engine uses cosine similarity to the query and an additional, selectable, scoring mechanism to rerank the results (by multiplying the similarity score with the score suggested by the selected quality measure). *.cs.umn.edu domain is crawled a few months after a major change in the site structure, allowing time for the site to become fairly stable. Approximately 6 months worth of usage logs starting from April 2002 are extracted around that snapshot. HTMLs are processed to build the static linkage graph. Snippets, titles, and links are extracted for each document. Logs are processed in order to obtain the usage based graph using both simple and modified counting approaches. The final data set contains a total of 65K URLs, out of which, about 20K URLs belong to the core pages for which we have extensive usage information (www.cs.umn.edu and www-users.cs.umn.edu domains).

UPR (Section 2.3.1) parameters a_1 and a_2 are sampled in increments of 0.25 from 0 to 1. Thus, a total of 25 different combinations are calculated using modified counts approach. The simple counts approach is also tried for a subset of these combinations to examine the effects of the modification. Note that the combination corresponding to $a_1 = a_2 = 0$ is exactly the same as PageRank in both cases. Various damping factor values have been tried. Values around 0.75 to 0.9 performed similarly and reasonably well. In order to facilitate comparison with other work in the field, the damping factor, d, is set to 0.85 for all PR and UPR variations throughout the reported experimental results. Note that this value has been suggested by the original PageRank paper [37], and has also been used



data mining

Results: 50 Method: UPR Site: CS

The query *data mining* produced **295** hits in **0.01** seconds.

1	Vipin Kumar's Home Page.. Vipin Kumar's Home Page.. Vipin Page.. Affiliations and Address: Director , Army High Performance Computing Research Center , kumar@cs.umn.edu http://www. mar/ Department of **1.00** *http://www-users.cs.umn.edu/~kumar/* 0.000139727 cach
2	**Data Mining** Bibliography Knowledge Discovery and **Data Mining** Bibliography (Gets updated frequently, so visit often!) Books General **Data Mining** Classification Association Rules Clustering **Mining** Sequential (or Temporal) **Data** **0.95** *http://www-users.cs.umn.edu/~mjoshi/dmrefs.html* 9.5746e-05 cached
3	Publications Related to **Data Mining** **0.73** *http://www-users.cs.umn.edu/~karypis/publications/data-mining.html* 5.38642e-05 cached
4	Research Papers Following are PostScript files containing papers by the parallel computing research group of Vipin Kumar . **Data Mining** Graph Partitioning Parallel Solution of Sparse Linear System of Equations N-Body **0.61** *http://www-users.cs.umn.edu/~kumar/papers/papers.html* 7.94512e-05 cached
5	Mahesh Joshi's Research Work Academic and Professional Chronograph November 2002: Doctor of Philosophy in Computer Science at University of Minnesota , Minneapolis. (Advisors: Prof. Vipin Kumar and Dr. Ramesh Agarwal, IBM **0.40** *http://www-users.cs.umn.edu/~mjoshi/myw.html* 5.66289e-05 cached
6	Special Issue on **Data Mining** in Science Postdoctoral Positions for Research on Parallel Algorithms Applications are invited for postdoctoral positions jointly by the department of Computer Science and Mechanical Engineering at the **0.27** *http://www-users.cs.umn.edu/~karypis/postdocs/index.html* 3.73093e-05 cached
7	Publications Related to **Data Mining** **0.26** *http://www-users.cs.umn.edu/~karypis/publications/ir.html* 3.76836e-05 cached
8	Research Problems for Spatial Databases Research Problems for Spatial Databases Research Problems for Spatial Databases This docuement has two sections. Consider scrolling down to appropriate sub-section if the entire document is not of **0.25** *http://www-users.cs.umn.edu/~shekhar/research/researchQ.html* 3.42269e-05 cached
9	Publications Related to **Data Mining** **0.25** *http://www-users.cs.umn.edu/~karypis/publications/bioinform.html* 3.21582e-05 cached
10	**Data Mining** Context Next: Comparison with some of Up: Experimental Results and Discussion Previous: Graph Theoretical Context **Data Mining** Context Here the results are presented in the **data mining** context. For all the

Figure 4.1. USearch user interface

in subsequent literature.

Since modifying HITS via usage statistics was suggested previously, original HITS as well as the proposed, generalized version of HITS modified via usage statistics (UHITS, Section 2.3.2) are implemented. The parameter space for UHITS is also sampled in increments of 0.25. Thus, in total, 5 hub score vectors, and 5 authority score vectors are calculated. Again, both hub and authority score vectors when $a = 0$ are equivalent to their original HITS counterparts.

Two versions of the naïve approach (Section 2.3.3) are also implemented (Counts and MCounts). In the first version, simple counts of number of visits to pages are used, i.e. each page is assigned a score directly proportional to the number of times users visited the page. The second version is similar: Analogous to UPR simple/modified counts approaches, instead of simple counts, log transform of the counts (Eq. 2.11) aggregated through different time windows are used.

Table 2.1 summarizes the characteristics of the three usage based methods in terms of types of information they use. Note that, in the case of UHITS and UPR, if the usage emphasis parameter(s) are set to exactly 0 or 1, some of the information types are effectively ignored.

All of the methods and variations discussed above add up to 37 different quality measures (score vectors). In Section 4.2, UPR is compared against PR by examining global scores and orderings under varying usage emphasis factors. The effects of the simple counting vs. the modified counting schemes are also examined in detail. Then, in Section 4.3, all score vectors are examined in terms of global correlations among each other as well as the distinguishing power they offer. Finally, in Section 4.4, a representative of each method is compared against others in a query dependent setting via explicit user evaluations, focusing on the overall quality of these methods aggregated through multiple queries.

In the following sections, Counts and MCounts denote the naïve quality measures using simple and modified counts, PR denotes PageRank, UPR(a_1, a_2) denotes UPR with parameters a_1 and a_2, UPR(a) denotes UPR with parameters a_1 and a_2, both set to a, and UHITS(a) aut/hub denotes HITS authority/hub vectors with usage emphasis set to a.

4.1 Characteristics of the Dataset

The domain/site examined in this study, *.cs.umn.edu, is maintained by the Department of Computer Science, University of Minnesota. Bulk of the departmental home pages, various information sources/manuals, and some of the research projects/groups' home pages are hosted on www.cs.umn.edu. User homepages are mostly separated from the main www.cs.umn.edu servers by a permanent redirect from www.cs.umn.edu/˜username to the corresponding location on www-users.cs.umn.edu. Apart from these two major sub-domains, there are various smaller sub-domains for different projects and research groups. www.cs.umn.edu, and www-users.cs.umn.edu are maintained by the department of Computer Science staff. Majority of the other domains are hosted on different servers mostly maintained by the group they belong to. Extensive usage information via full server access logs were available for www.cs.umn.edu and www-users.cs.umn.edu.

The final data set did not contain secure content, password protected portions, and the public portions that were removed upon request by their owners. For these two sub-domains, more than 20K mostly static HTMLs were included in the final data set. Some of the dynamically generated HTMLs, and other types of content that are not trivial to process/index (e.g. PS/PDF documents, non-HTML presentations) were not included.

Usage statistics for most of the other sub-domains within *cs.umn.edu were unavailable at the time the experiments were carried out. The data set contains peripheral pages within these domains, as well as in other related domains within *.umn.edu, such as itlabs.umn.edu. Apart from www.cs.umn.edu and www-users.cs.umn.edu, these peripheral pages/domains brought in an additional 45K+ static HTML pages. Reference to these pages and partial usage information (seen through the available server logs) are used within some of the algorithms as they provide valuable information. However, they are not included within the score vectors, and they are not searchable from USearch.

4.2 Comparing PR vs UPR

4.2.1 Effects of Usage Emphasis

In the first set of experiments, the simple counting approach is used for obtaining the usage graph (IP address and time stamp fields are ignored). As expected, depending on the emphasis factor between the static structure graph and the usage graph, scores and relative ordering of pages deviate from PageRank (PR) as usage emphasis is increased. www.cs.umn.edu contains a number of online manuals and information pages for various applications and software, forming quite large "sub-sites" with relatively high connectivity. For instance, every page in the manual may have a link to the starting page of the manual as well as cross-links. However, most of these pages are hardly accessed. Without introducing the usage graph, most of these pages as well as important pages for presentations, discussion boards etc. dominates the top 100 positions using regular PR. For example, out of 20 top ranked pages, 5 were Cisco documentation pages, 6 were Java JDK documentation pages, and 2 were FAQ pages for online class discussions pointed by most of the class web pages. Various Matlab documentation pages were also among the top 100 highest ranked pages. The department's main page itself, www.cs.umn.edu/, was ranked 136^{th} in terms of pure PageRank score.

As soon as usage information is introduced, department's main page, as well as users' and research groups' home pages start to rise in rank. The emphasis value is sampled in increments of .25 from 0 (pure structure based) to 1 (pure usage based). Both a_1 and a_2 are set to the same value in order to limit the number of samples that needs to be examined manually. Figure 4.2 shows the distribution of scores of all pages in log scale for these values of the parameters. Note that the core 20K documents are placed in the left portion of the figure as they have higher connectivity as well as higher usage. The remaining URLs are peripheral pages, or pages that are pointed by/point to the core but are hardly pointed by other pages within the data set. The positions and scores of major pages with different emphasis values are examined further. Using an emphasis value of .25, department's home page is ranked 6^{th}, and using a value of .5 and higher, it becomes the highest ranked page. Note that, for a value of 0 corresponding to regular PageRank, it was in the 136^{th} position.

On the other extreme, as the emphasis on the usage graph is increased, a few pages start to be ranked unintuitively high. In particular, using usage graph only (UPR(1)), the 2^{nd} and the 3^{rd} highest scored URLs are:

- www-users.cs.umn.edu/~*userone*/ip/

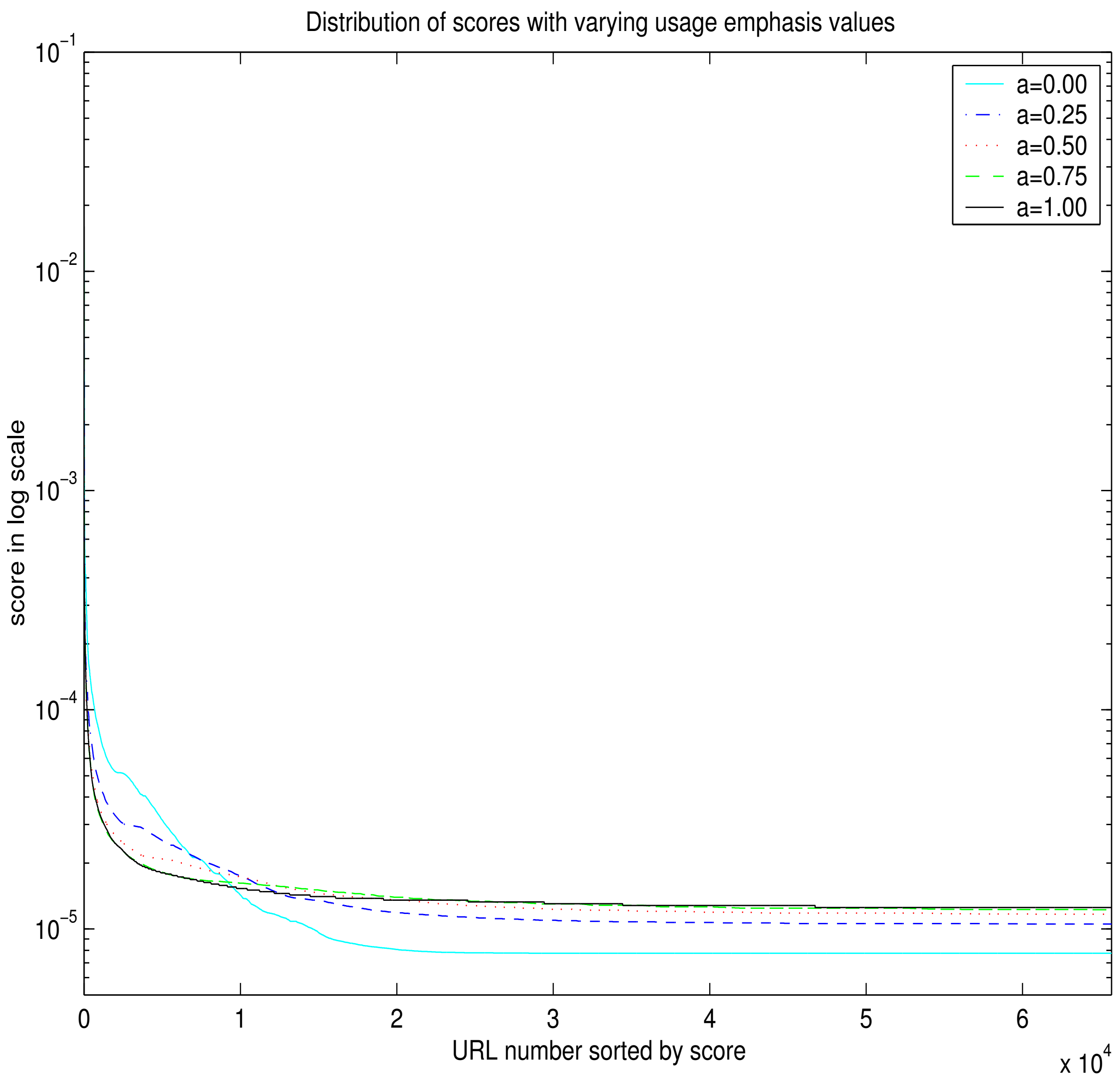


Figure 4.2. Log UPR scores of pages using simple counting scheme for different emphasis values between static and usage based graphs

- www-users.cs.umn.edu/~*usertwo*/links.html

(Usernames are omitted for anonymity). These two pages are placed at the 6th and the 10th positions for emphasis factor of 0.75 (a_1=a_2=0.75), and at the 10th and the 18th positions for emphasis factor of 0.5). There are significant differences between the PR and UPR(1) scores of these pages. Moreover, none of the two pages intuitively seem as important as other pages observed in top ranks. Upon further examination, it turned out that the first page was a simple page, consisting of an IP number, the dynamic IP number of that user's home computer, uploaded by a script when it changes. This page was accessed regularly to obtain the latest IP address of the user's home computer. The second page was a graduate student's home page containing search boxes for a number of search engines as well as various local and global links. We found out that the student and his colleagues had the page bookmarked as their browsers' home page, generating hits every time they open up a web browser or issue a query. Other pages that are investigated in top positions were mostly intuitive and subjectively important pages.

Next, focusing on the core set, the two extremes are compared by dividing the score of each page using emphasis value of 0 by the score of the same page using emphasis value of 1 (pure static vs. pure usage). The list is then sorted according to this ratio. The higher values (above 1) correspond to the pages that have relatively higher scores using the static structure graph; the lower values (below 1) correspond to pages that have relatively higher scores using the usage graph. The ratio for the whole collection of pages ranged from 0.0062 to 895.5. As a reminder, scores in all implemented methods are L1 normalized and add up to 1. As expected, pages having the highest ratios are mostly the manual pages. In fact, all the pages having a ratio above 100, except one, are manual pages and two www board pages pointed to by almost all pages in various discussion boards. In contrast, the pages that are in the bottom portion of the list were mostly department's main pages, user home pages, and research projects' home pages. The two anonymized URLs mentioned previously, as well as www.cs.umn.edu/, www.cs.umn.edu/users/, and www.cs.umn.edu/classes/ are among the bottommost 20 pages. An interesting trend is that the bottom portion of the list was mostly populated by many URLs having a "~" (tilde) character inside, suggesting that these are mostly user home pages (389 out of 500 pages that are investigated). Whereas the reverse was clearly visible for the top portions of the list (only 79 out of 500 pages had a tilde character).

4.2.2 Effects of Modified Counting Scheme

The experiments are repeated using the modified counting scheme while building the usage graph, again, sampling the emphasis parameter in increments of 0.25, and setting both a_1 and a_2 to the same value. In general, results obtained using the modified counting scheme are very similar to the ones using the simple counting scheme. Overall order of majority of the pages remained stable, except for a small portion of pages, which are typically accessed by very few users, but a large number of times. For majority of the URLs, it was possible to find the same URL around the same positions in the other list. The new distribution of scores is shown in Figure 4.3, which is almost identical to Figure 4.2 (simple counting scheme). Department's main page was again in the 6^{th} position for 0.25, and in the 1^{st} position for values of 0.5, 0.75 and 1.0. In fact, the dot product (when L2 normalization is used), Pearson correlation, as well as Spearman correlation of UPR(1.0), UPR(0.75), and UPR(0.5) vectors using simple counting scheme vs. their counterparts using modified counting scheme are all above 0.99. On the other hand, unlike most of the pages, positions of the two pages mentioned in the previous case: www-users.cs.umn.edu/˜*userone*/ip/ and www-users.cs.umn.edu/˜*usertwo*/links.html are quite different in the corresponding new lists. Using the modified counting scheme, they are placed in positions 130^{th} and 40^{th} for emphasis value of 1.0, 180^{th} and 67^{th} for 0.75, and 329^{th} and 116^{th} for 0.50. Note that, in the previous case, they were in positions 2^{nd} and 3^{rd}, 6^{th} and 10^{th}, and 10^{th} and 18^{th} respectively.

Full usage based scores (a_1=a_2=1.0) produced by simple counting vs. modified counting approaches are compared by dividing the UPR of a given page using simple counting scheme by the UPR of the same page using modified counting scheme. The distribution of the ratios are shown in Figure 4.4. The middle portion of the graph is almost flat and close to 1. Figure 4.5 further focuses on the first 500 and the last 500 URLs for better visibility, removing the flatter portion. The modified counting scheme does not change the overall order or scores of most of the URLs, but it helps filter out URLs that are accessed by very few people many times. It also makes the algorithm less spam prone, i.e. it would be harder for a user to boost the UPR of a page artificially by generating large amount of traffic from a single source or a few sources.

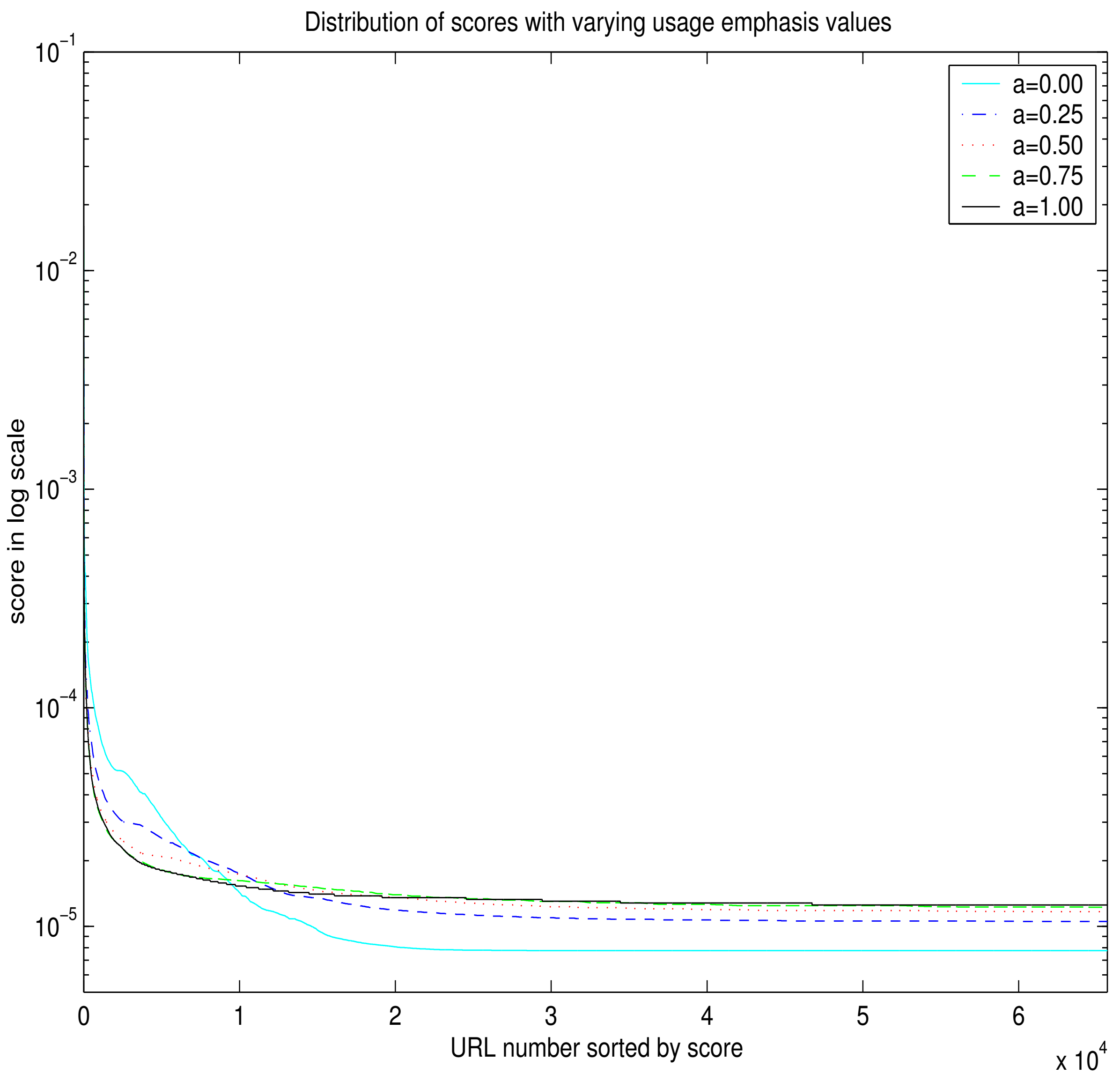


Figure 4.3. Log UPR scores of pages using modified counting scheme for different emphasis values between static and usage based graphs

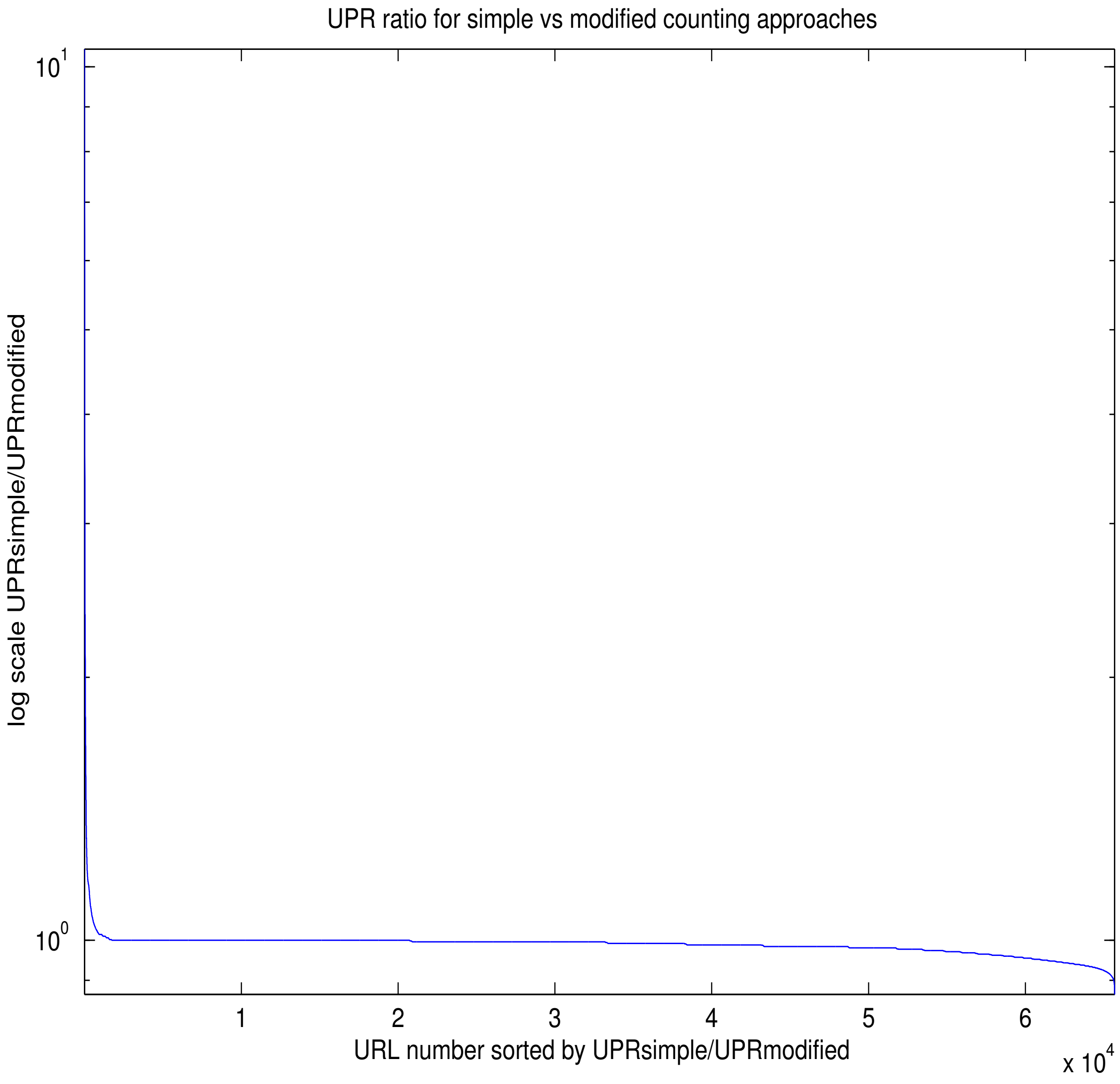


Figure 4.4. Comparing UPR using simple counts vs. UPR using modified counts

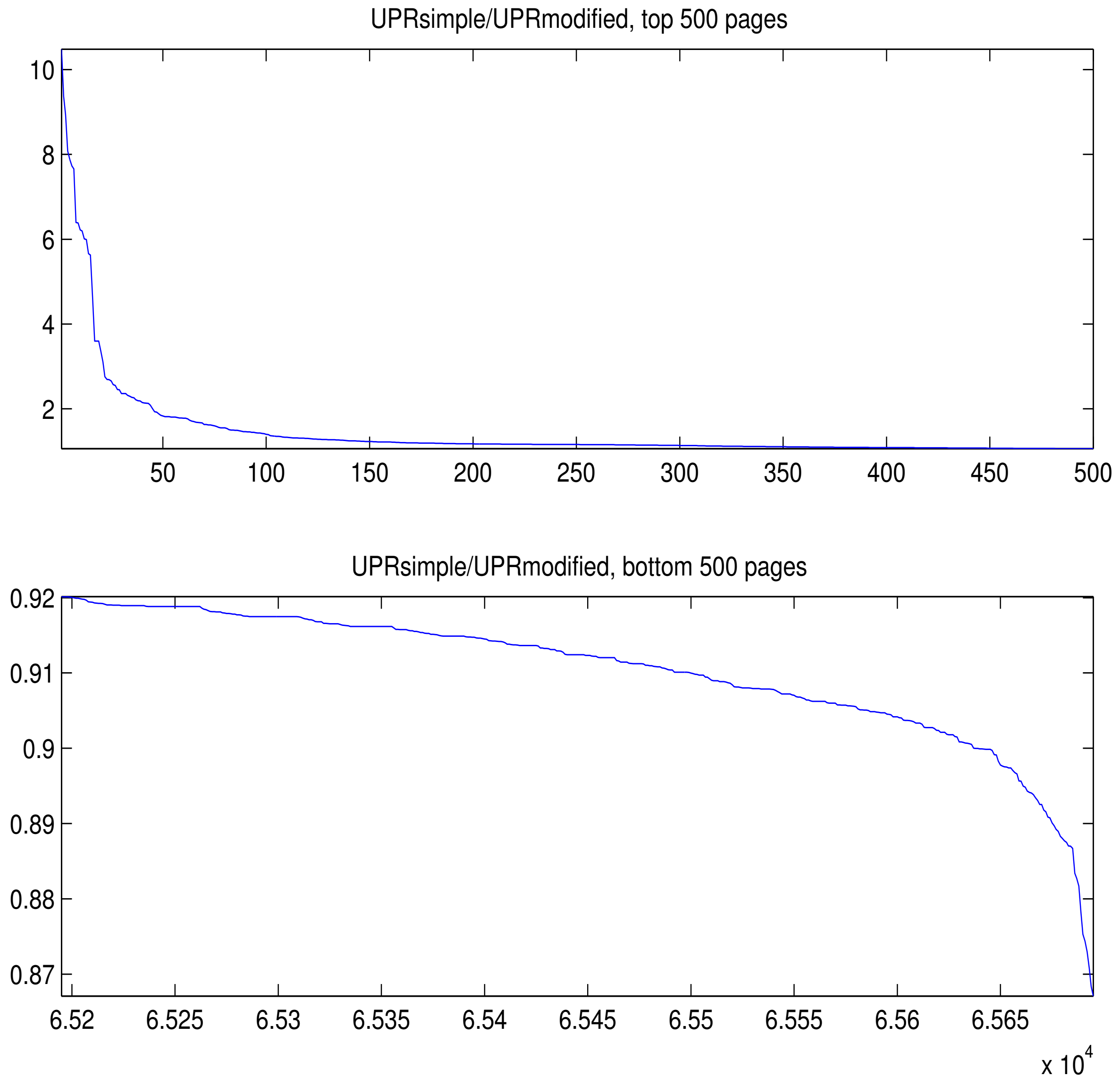


Figure 4.5. Comparing UPR using simple counts vs. UPR using modified counts, focusing on top and bottom 500 pages

4.3 Pairwise Comparison of Quality Measures

In this experiment all score vectors are compared against each other in terms of global orderings, the distinguishing power they offer, and how smooth/sharp the behavior of a method is when the parameters are changed. Focusing on the 20K core URLs corresponding to www.cs.umn.edu and www-users.cs.umn.edu domains, for each method pair, the following metrics are calculated:

- Pearson correlation
- Spearman correlation
- second norm of the difference vector
- cosine similarity (dot product)

Pearson correlation compares the score vectors produced by two methods and is defined as:

$$\frac{\sum_{i=1}^{n}((x_i - \bar{x})(y_i - \bar{y}))}{(n-1)S_x S_y} \tag{4.1}$$

where x and y are two variables (scores produced by methods x and y) with means $\bar{x}$ and $\bar{y}$, and standard deviations S_x and S_y respectively.

Spearman correlation uses rank order instead of scores and can be calculated using Equation 4.1 when scores are replaced by ranks. Pearson and Spearman correlations range from -1 to 1; values closer to 0 suggesting poor correlation, and values closer to 1 (-1) suggesting stronger positive (negative) correlation.

Provided that the vectors are L2 normalized, the dot product of two vectors x and y compares their direction. For L2 normalized vectors, it reduces to the following formula:

$$\sum_{i=1}^{n}(x_i y_i) \tag{4.2}$$

Note that the possible range for L2 normalized vectors with non-negative entries is $[0, 1]$ (1 when the two vectors have the same direction i.e. perfect similarity, 0 when the vectors are orthogonal).

Second norm of the difference vector is zero when the two vectors are identical, and increases as they deviate from each other.

All score vectors are L1 normalized, except in the case of dot product, where L2 normalized scores are used. Figures 4.6, 4.7, 4.8, and 4.9 summarizes the results for each method pair for the above metrics. Ordering from left to right and top to bottom is as follows: rows/columns 1 and 2: Counts, MCounts (naïve approaches), 3: PR (same as UPR(0,0)), 3-7: UPR(a_1=0, a_2=0 to 1), 8-12: UPR(a_1=0.25, a_2=0 to 1), 13-17: UPR(a_1=0.5, a_2=0 to 1), 18-22: UPR(a_1=0.75, a_2=0 to 1), 23-27:

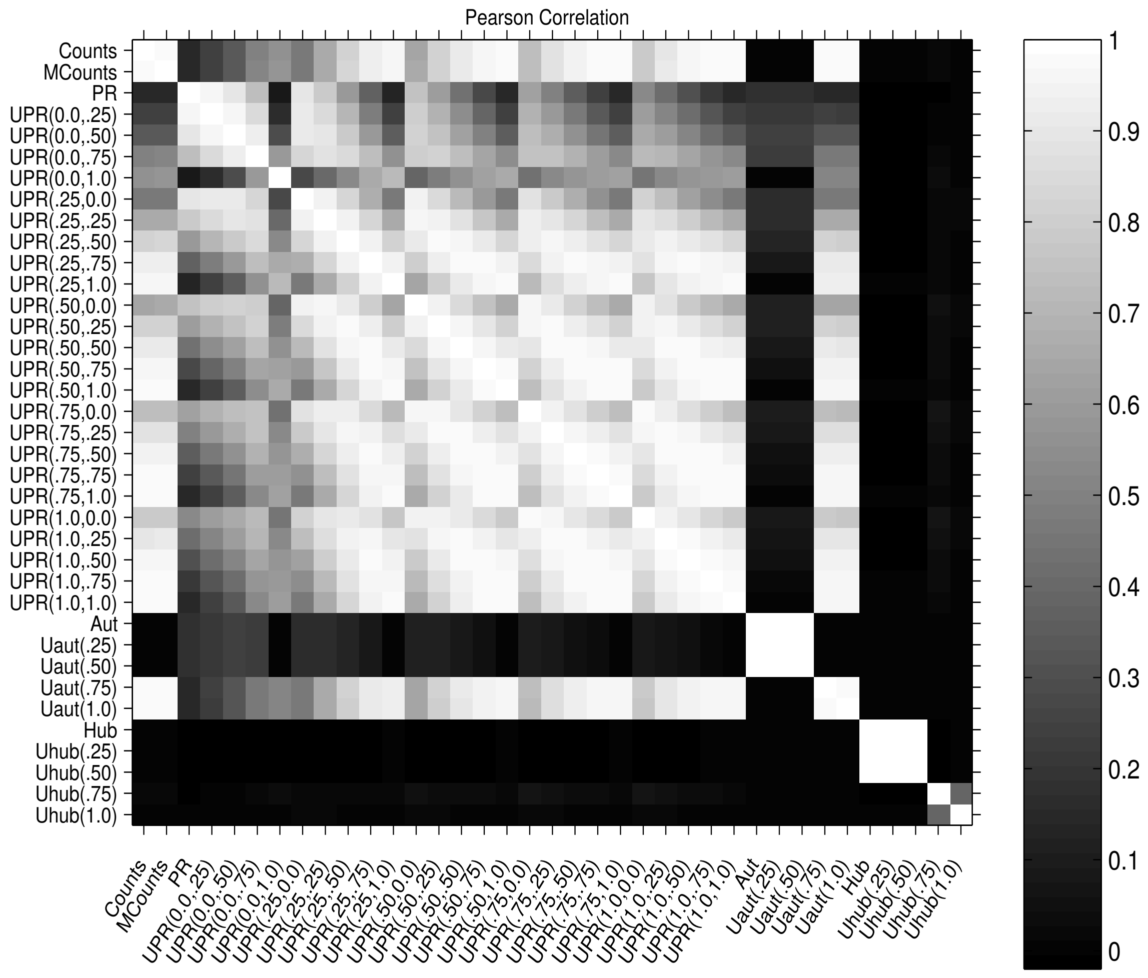


Figure 4.6. Comparing scores: Pearson correlations

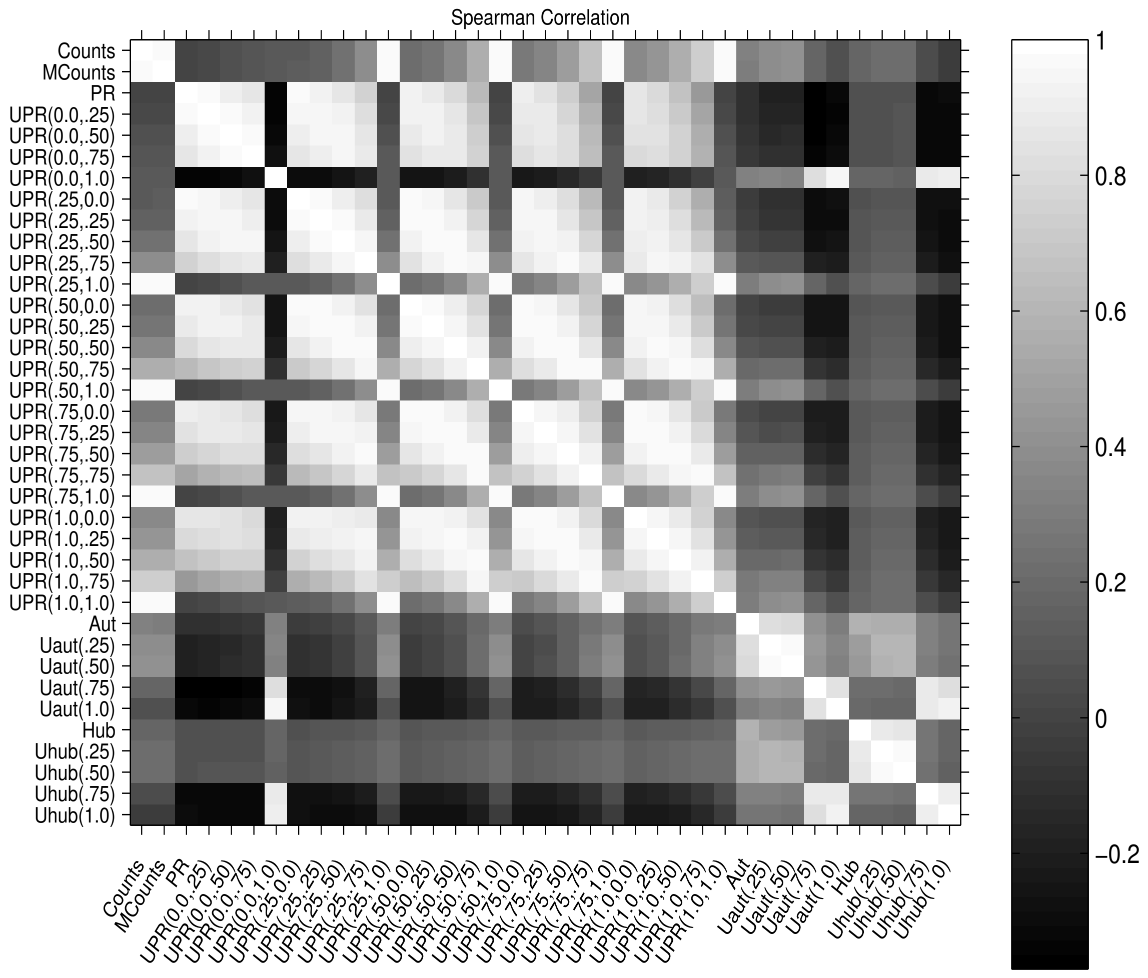


Figure 4.7. Comparing ordering: Spearman correlations

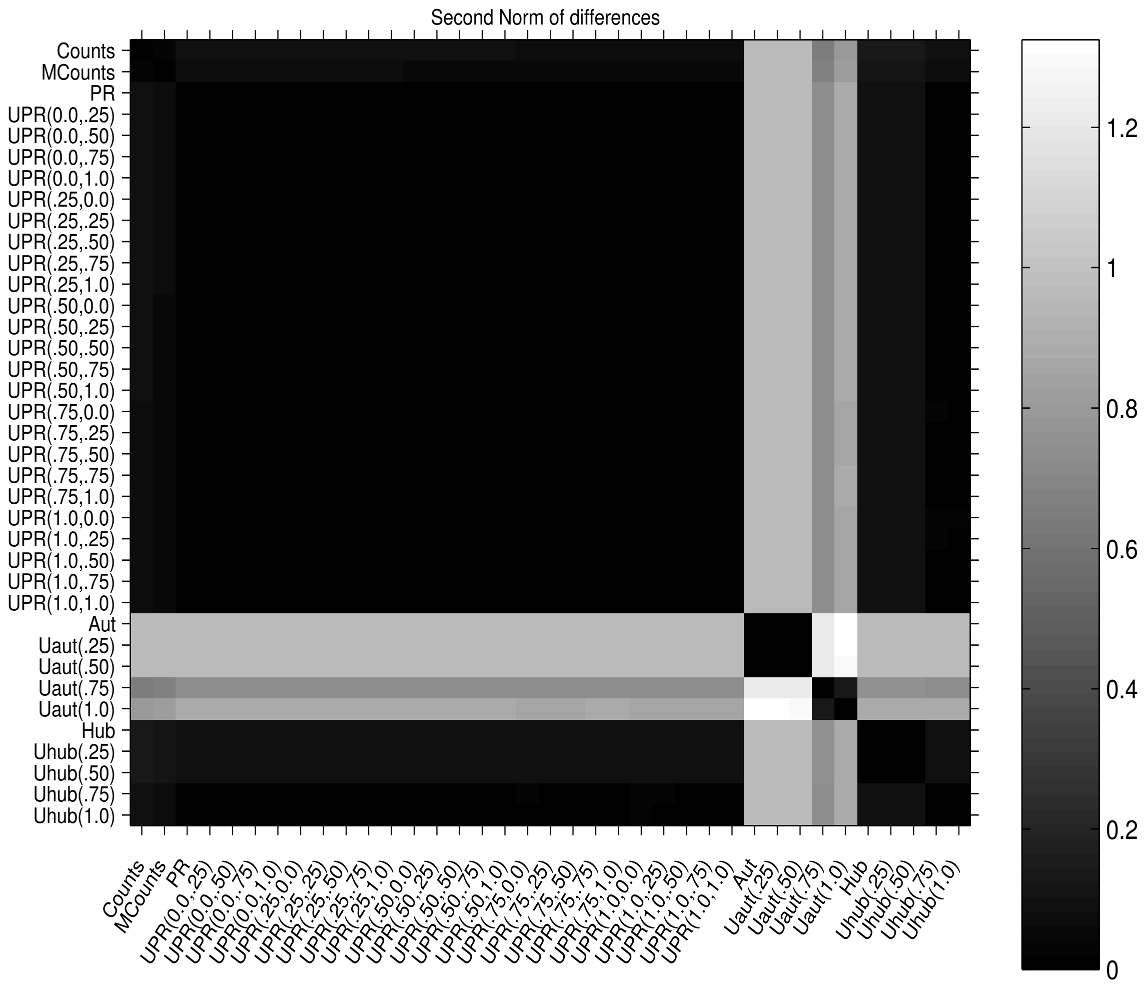


Figure 4.8. Comparing differences scores: Second norm of the difference vector

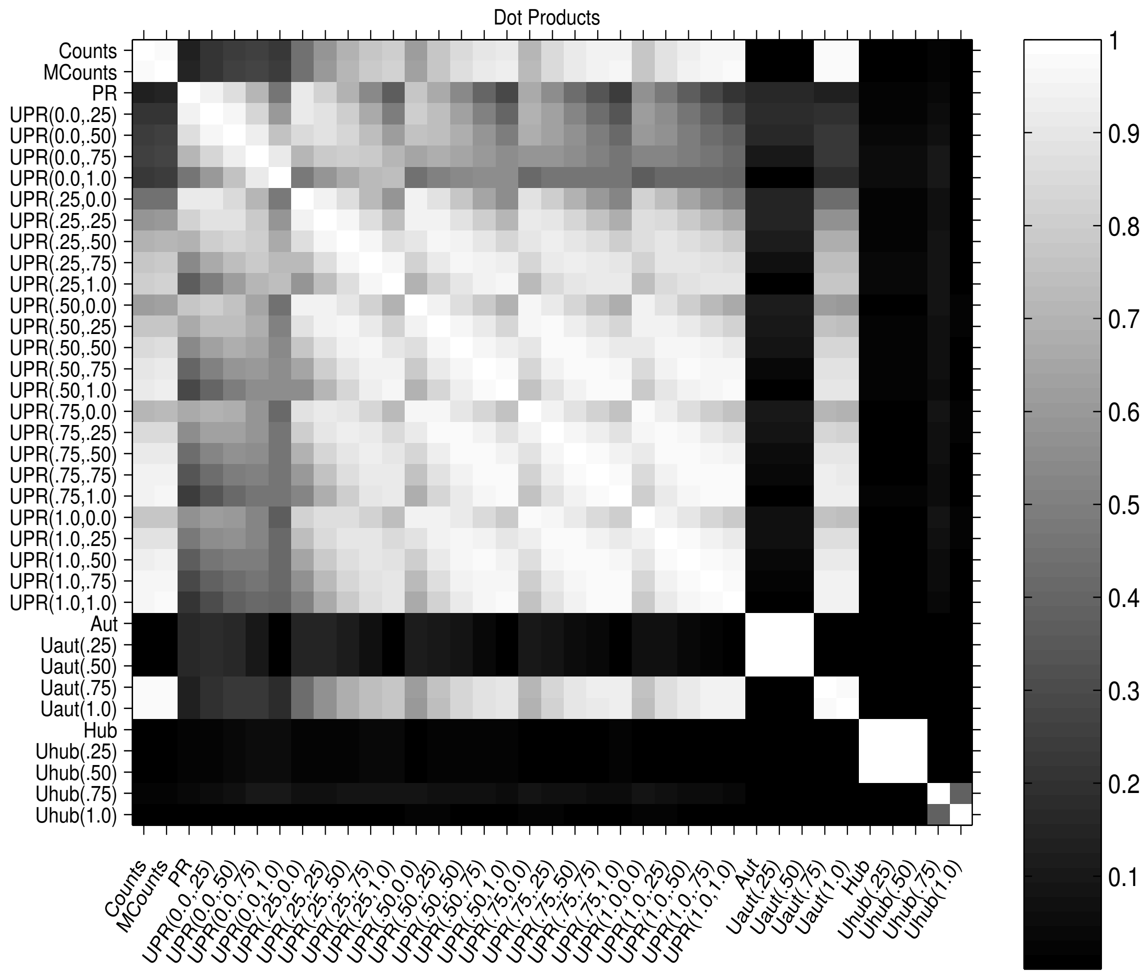


Figure 4.9. Comparing scores: Dot produtcs (cosine similarity)

UPR(a_1=1, a_2=0 to 1), 28: HITS authority (same as UHITS(0) authority), 28-32:UHITS(a=0 to 1), 33: HITS hub (same as UHITS(0) hub), 33-37 UHITS(a=0 to 1) hub.

Pearson correlation and dot product produces similar results. However, due to the nature of score distributions in link analysis in general, relying only on the correlations in terms of scores may be misleading. Pearson correlation and dot product effectively measures the correlation between highly scored pages, while majority of the remaining pages have degrees of magnitude smaller scores, practically negligible in comparison. Spearman correlation focuses on the ordering and provides a different perspective. Even highly correlated score vectors may result in relatively different orderings. For instance, UPR(0.5, 0.5) and Counts has a Pearson correlation of 0.96. However, the Spearman correlation is significantly lower: 0.38. This effect is also observed while issuing queries. In majority of the test queries, different methods suggested relatively different orderings, sometimes within the top 5 positions, although the correlations in terms of global scores were relatively high.

Distinguishing Power and General Trends

Figures 4.10, 4.11, and 4.12 show the distribution of scores for pure PageRank, UPR(0.5), and UPR(0.75) respectively. The y axis shows the score of the URL in log scale; the x axis shows the URL number sorted by descending order of scores. Focusing on the core 20K URLs, PR and UPR with all parameters except a_2=1 are fairly successful in distinguishing between pages and offering a smooth global ordering. One particular observation is that, the slopes of the distribution functions are practically non-zero for these quality measures, suggesting that the number of pages having the same or similar scores are relatively few. It is also observed that practically none of the core pages are assigned a score of zero. Note that the core pages tend to appear on the left portion of the figures as they have higher scores in general.

Figures 4.13 and 4.14 show the distribution of scores for the naïve approaches, Counts and MCounts respectively. Unlike PageRank variants, it is observed that the distributions are not as smooth as UPR or PR, and have a more step-like nature, especially for pages having a lower score.

Portions with a zero or close to zero slope in Figures 4.13 and 4.14 suggest that a relatively large number of documents were assigned the same or similar scores using the naïve approaches. Naturally, Counts/MCounts are also unable to distinguish between pages that are rarely used (zero or very close to zero scores for these portions).

Figures 4.15, 4.16, 4.17, and 4.18 show the score distributions of selected HITS variants: HITS authority, HITS hub, UHITS(0.75) authority, and UHITS(0.75) hub respectively. These Figures suggest that HITS and HITS modified via usage, were the worst methods in terms of distinguishing power between pages. Out of the 20K core pages, for some parameter values, scores of more than

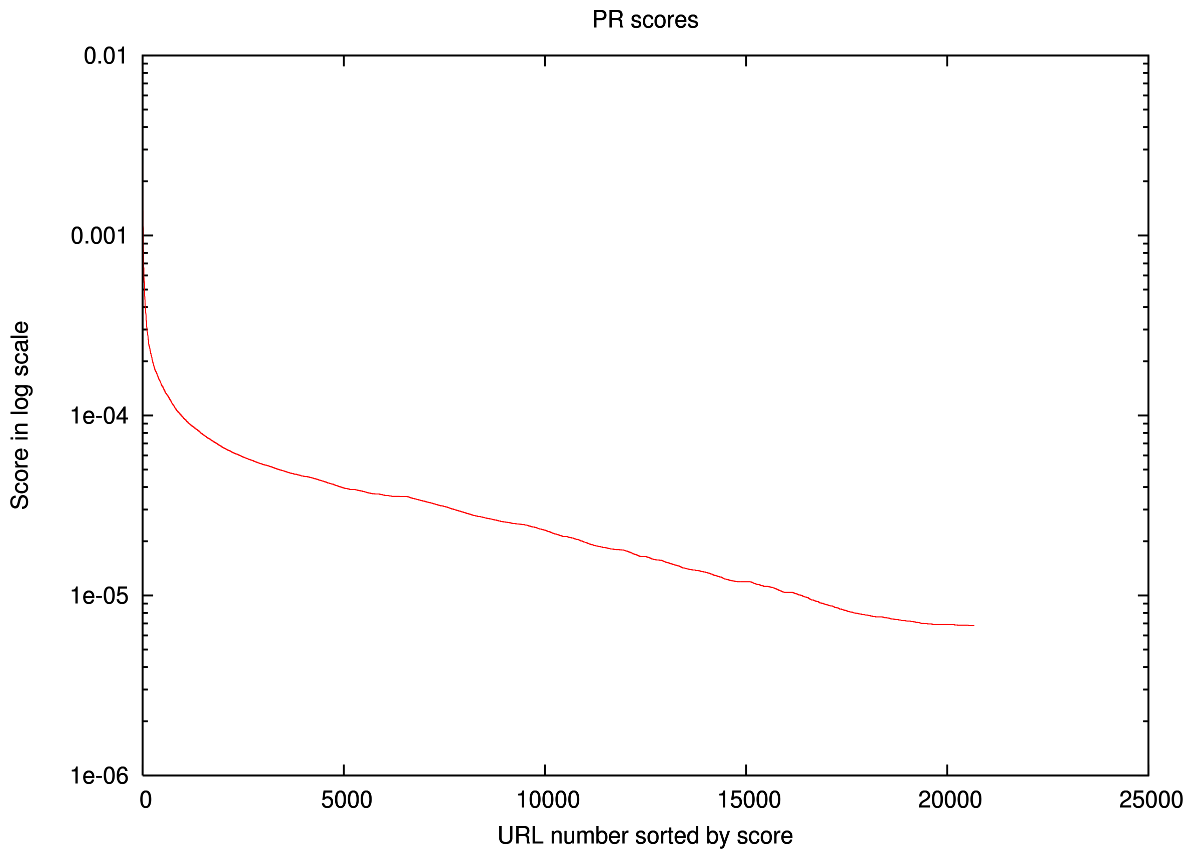


Figure 4.10. Distribution of PageRank scores

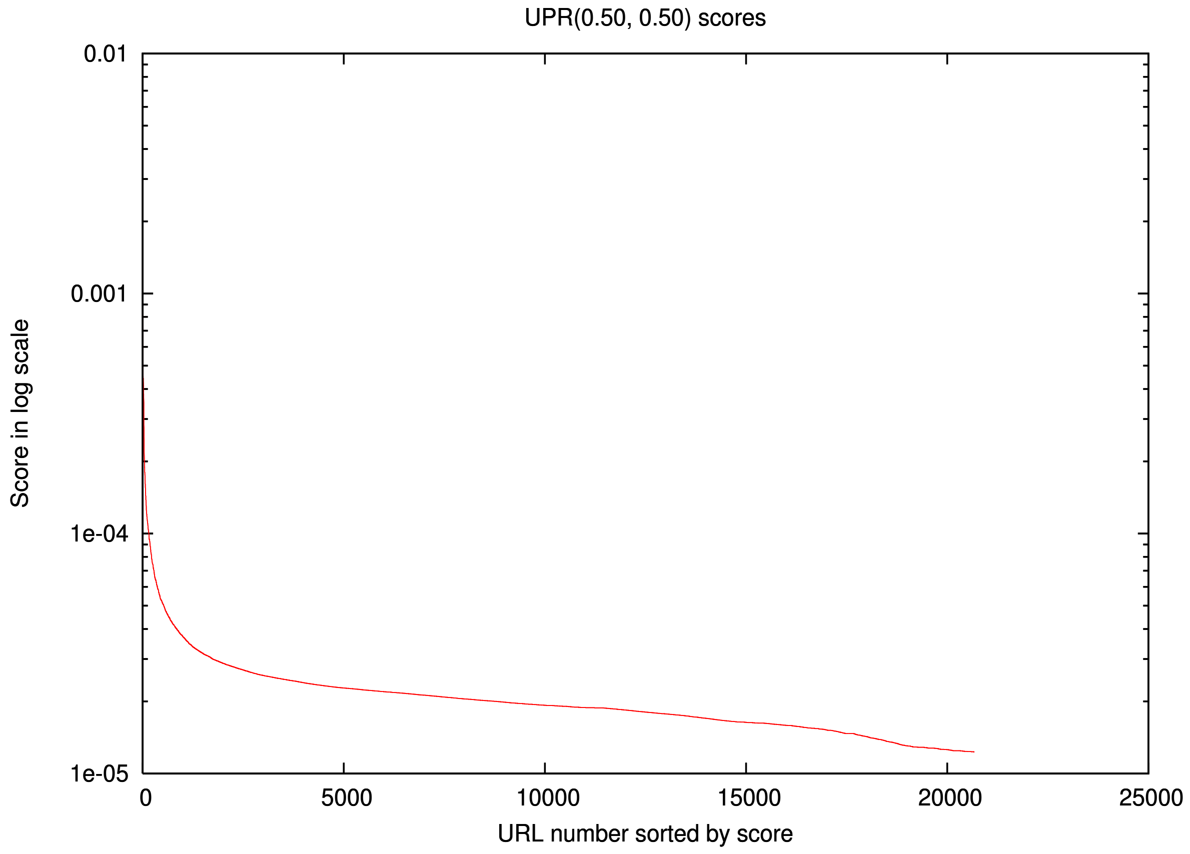


Figure 4.11. Distribution of UPR(0.5, 0.5) scores

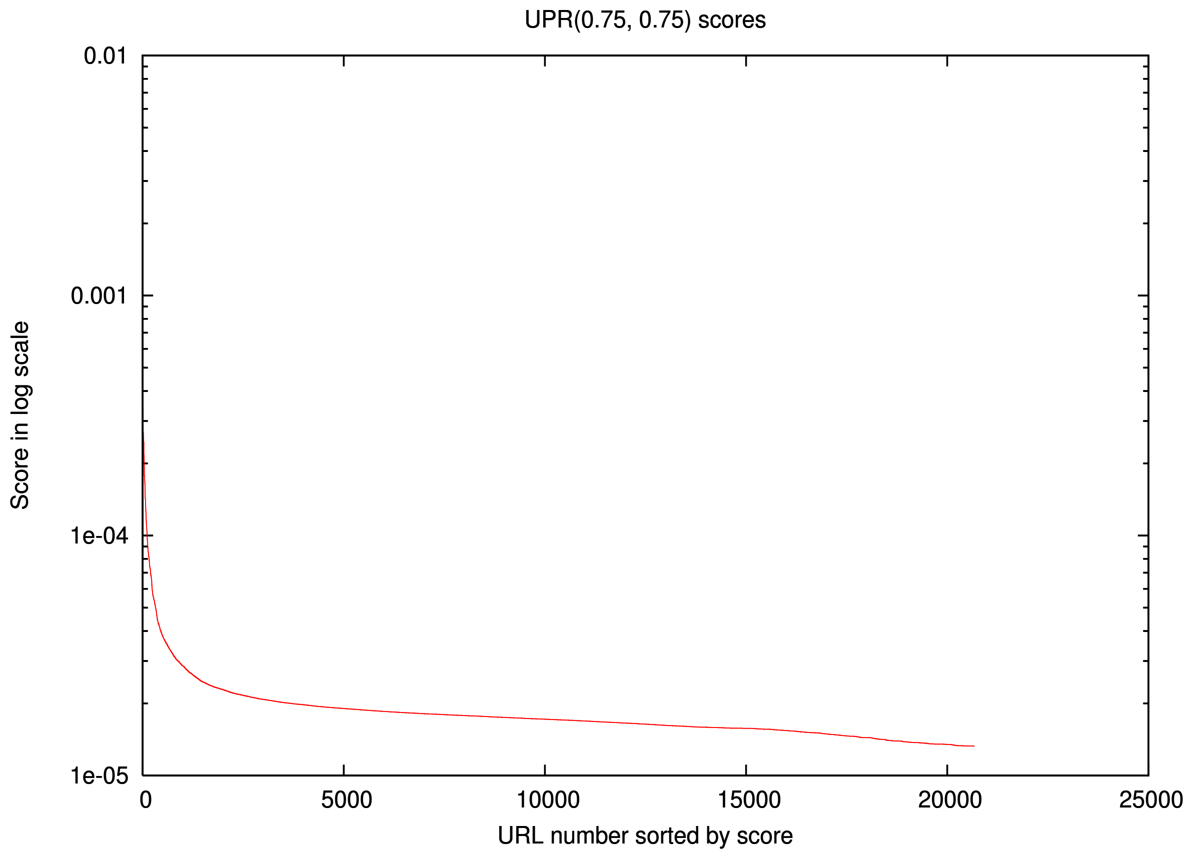


Figure 4.12. Distribution of UPR(0.75, 0.75) scores

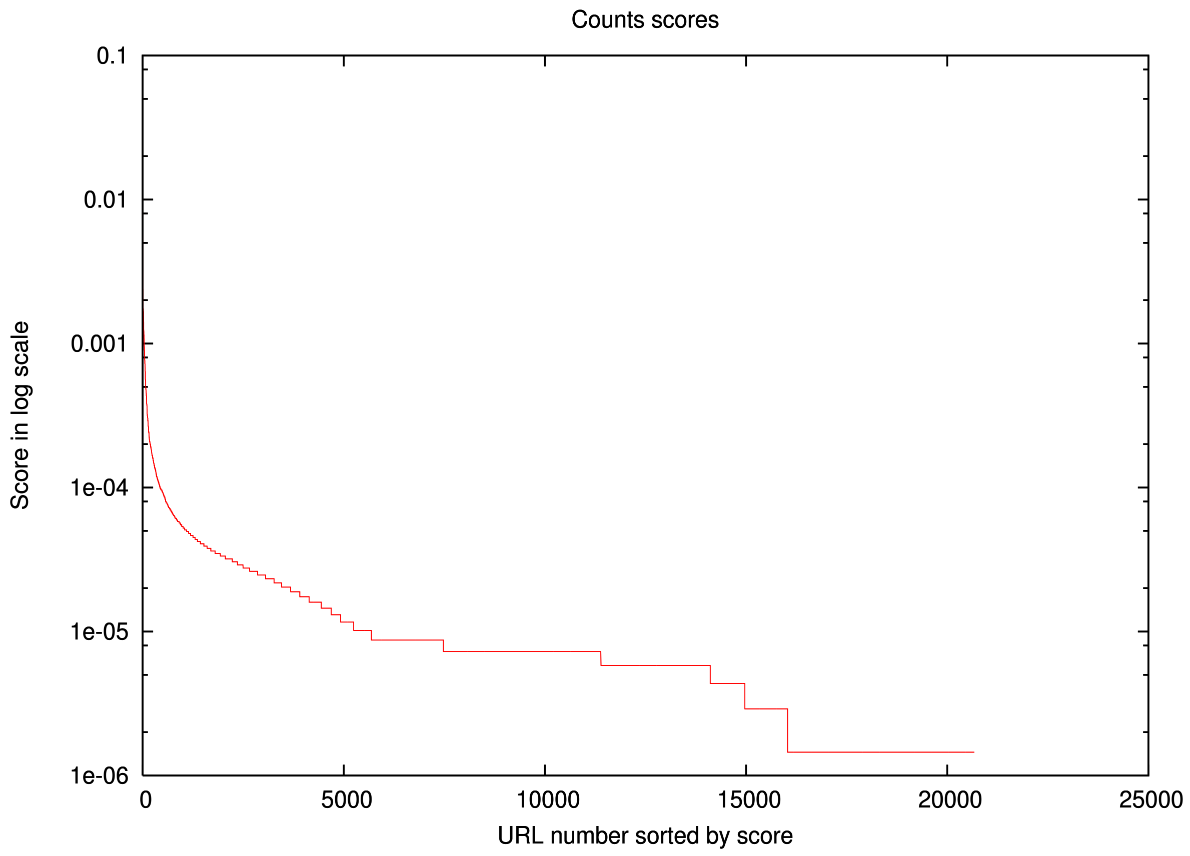


Figure 4.13. Distribution of Counts scores

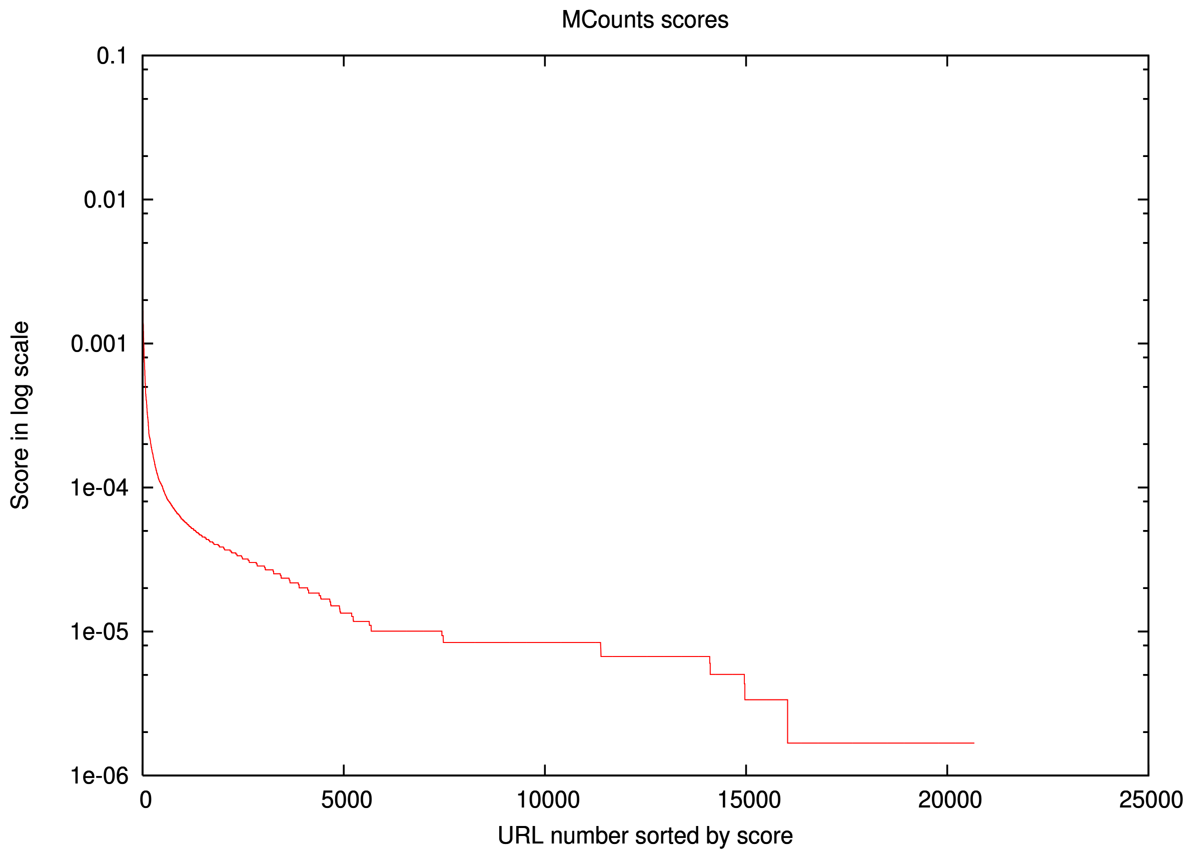


Figure 4.14. Distribution of MCounts scores

half of the pages converged to 0 using HITS variants. This effect is clearly visible in the case of UHITS(0.75) authority and hub scores (Figures 4.17, and 4.18).

Effects of Usage Emphasis Parameters

Figures 4.6 and 4.7, suggest that UPR behaves smoothly as the parameters a_1 and a_2 range from 0 to 1. For parameter values close to 0, the correlation between UPR and PR is quite high; for higher values, the correlation between UPR and the naïve approaches (Counts and MCounts) increases gradually. Figures 4.19 and 4.20 compares URP(0.25) with PR, and UPR(0.75) with MCounts respectively. The score of a document using one method is plotted against the score of the same document using the other method.

One interesting observation in Figure 4.20, which compares UPR(0.75) vs. MCounts, is the appearance of multiple horizontal (or close to horizontal) patterns. In these portions, MCounts suggests the same or similar score for these document clusters, while UPR(0.75) is able to offer a smooth range of scores. This observation is compatible with the individual score distributions (e.g. UPR(0.75) in Figure 4.12, and MCounts in Figure 4.14). The horizontal patterns can be easily attributed to the step-like portions of the MCounts score distribution (Figure 4.14).

No clearly visible horizontal patterns are observed while comparing PR and UPR(0.25) (Figure 4.19). This figure suggests that UPR(0.25) and PR shows reasonable correlation, especially in the denser region. It also suggests that neither of the two methods are clearly better than the other in terms of distinguishing power.

The behavior of UHITS authority as well as hub scores are not as smooth as UPR for varying parameter values. Figures 4.6 and 4.7 suggest an abrupt change in behavior while moving from UHITS(0.5) to UHITS(0.75), but for values below this threshold, it behaves almost identically (similarly, UHITS(0.75) and UHITS(1.0) have almost the same behavior). Also, except UHITS(0.75) and UHITS(1.0) authority vectors, HITS authority as well as hub scores are poorly correlated to other methods, sometimes showing negative correlation.

Figures 4.21, 4.22, and 4.23 compares PR with HITS authority, UPR(0.25) with UHITS(0.25) authority, UPR(0.75) with UHITS(0.75) hub, and Counts with UHITS(0.50) hub respectively. While comparing HITS/UHITS variants against PageRank/UPR variants, horizontal line patterns observed in the corresponding figures suggest that PR/UPR variants are better in distinguishing between pages for these document clusters (they were able to provide a range of values, while the HITS variatns offered the same/similar score). In general, while comparing PR and HITS variants, it is observed that the the data points are rather scattered, suggesting relatively lower correlations.

Figure 4.24, comparing Counts vs. UHITS(0.50) hub is of particular interest. Both horizontal

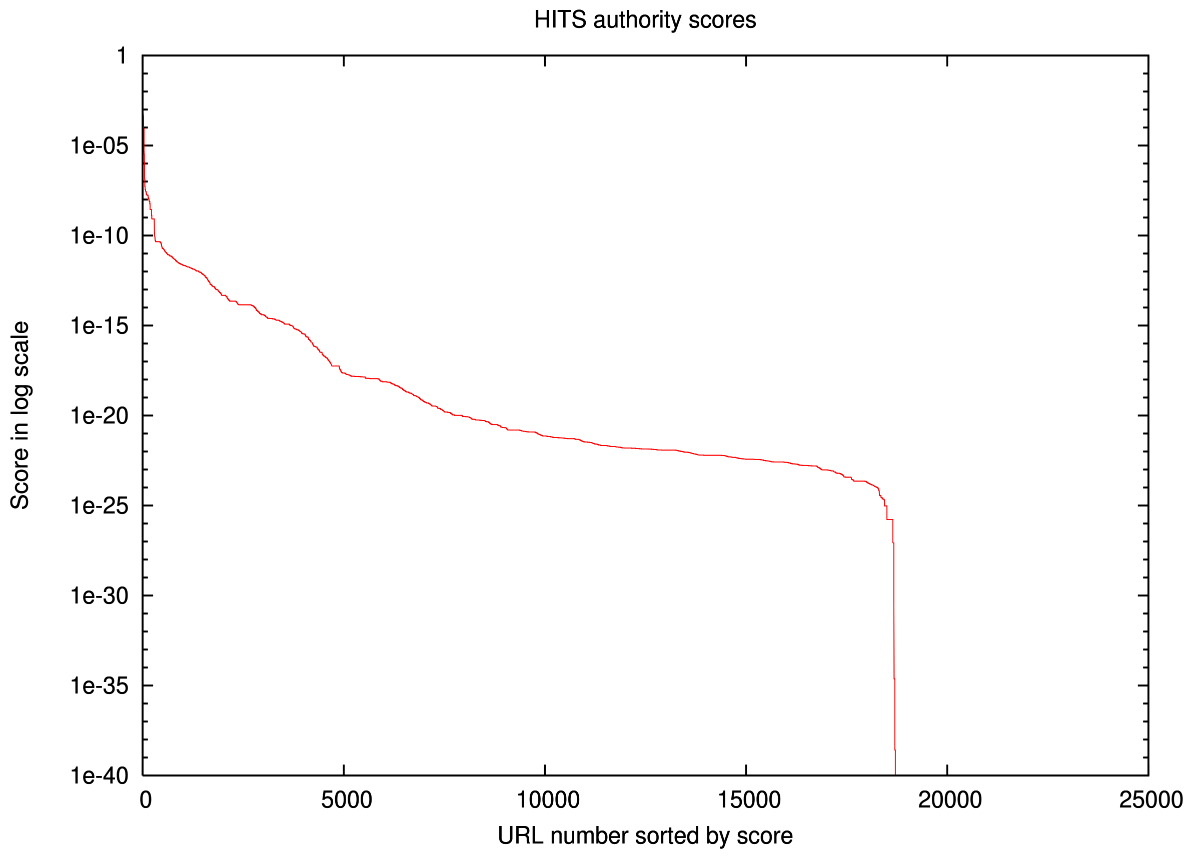


Figure 4.15. Distribution of HITS authority scores

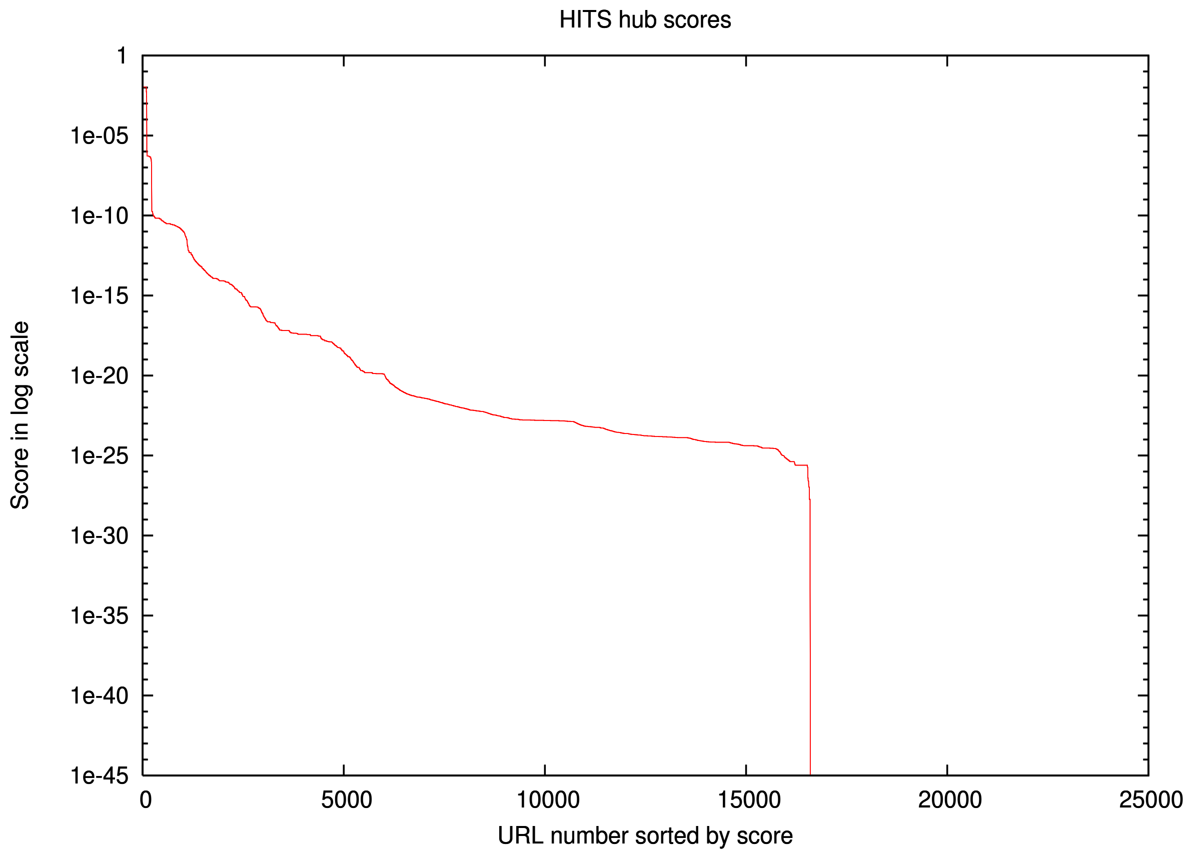


Figure 4.16. Distribution of HITS hub scores

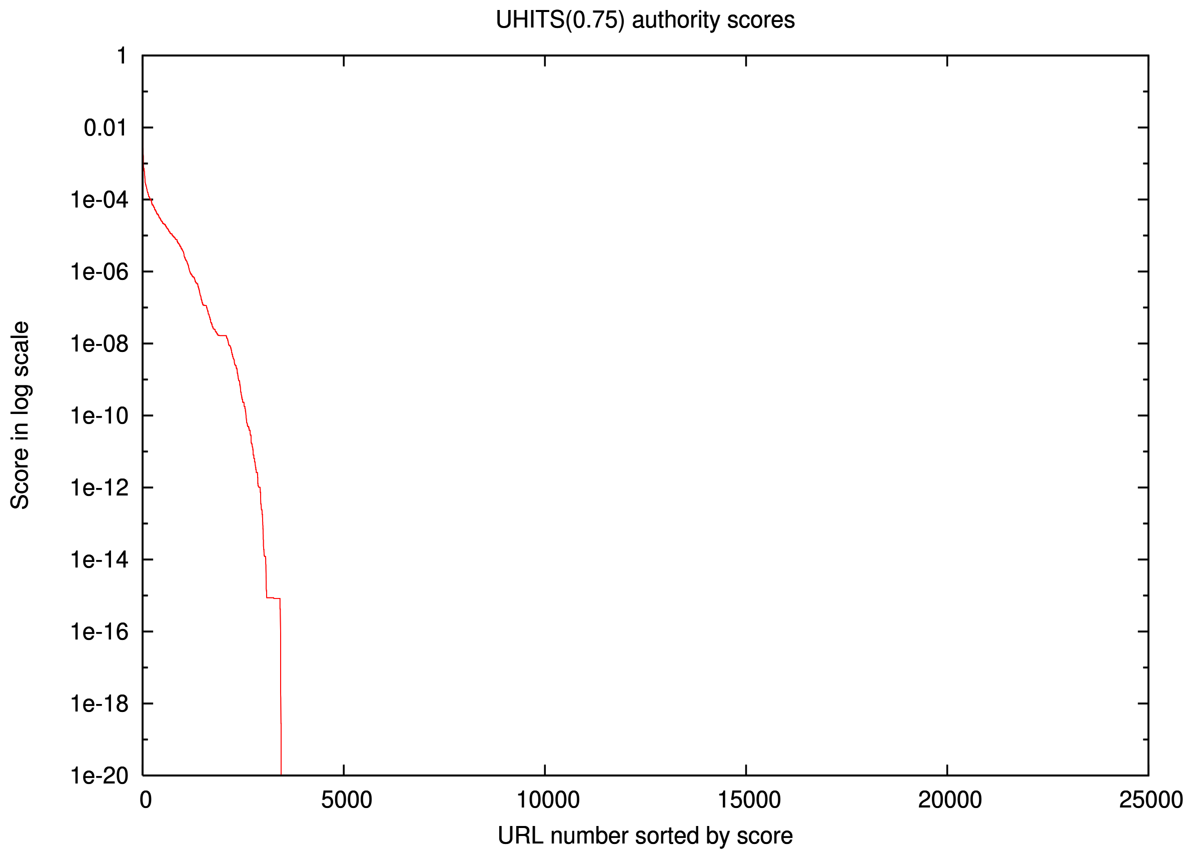


Figure 4.17. Distribution of UHITS(0.75) authority scores

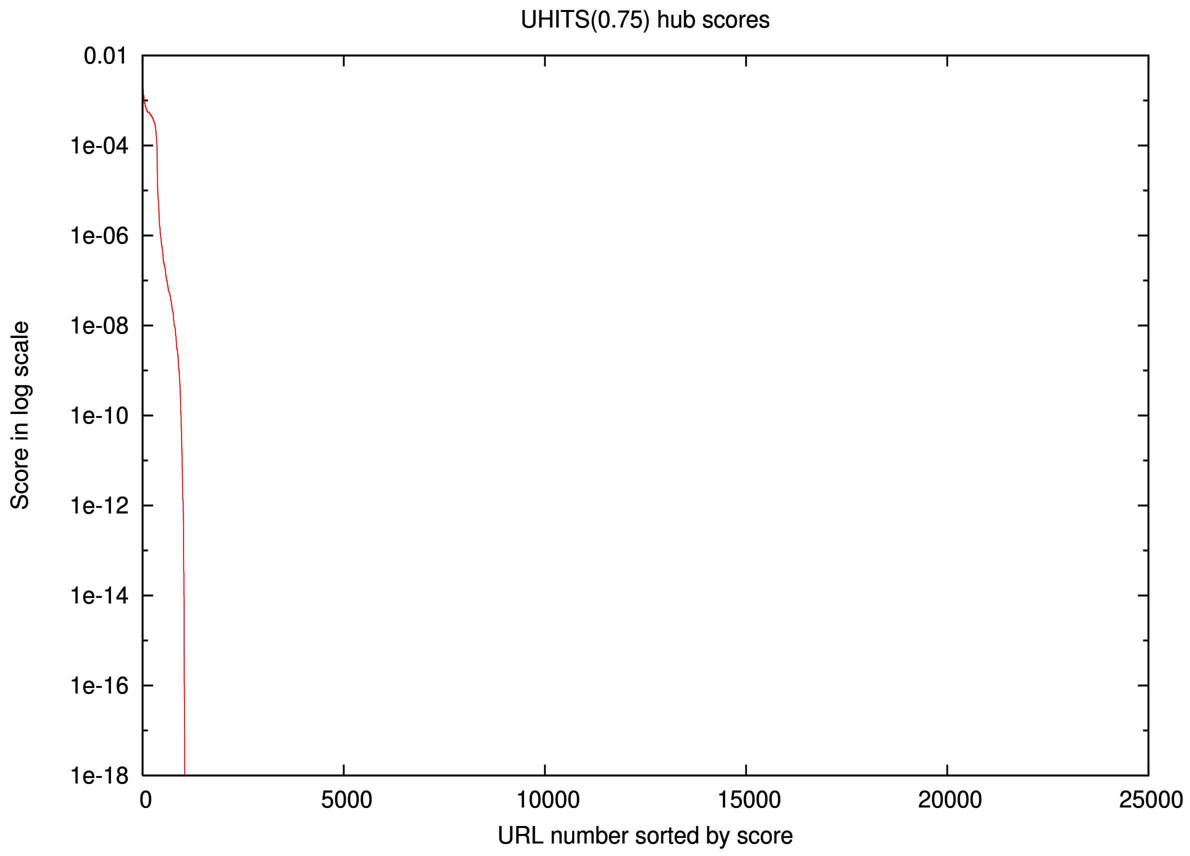


Figure 4.18. Distribution of UHITS(0.75) hub scores

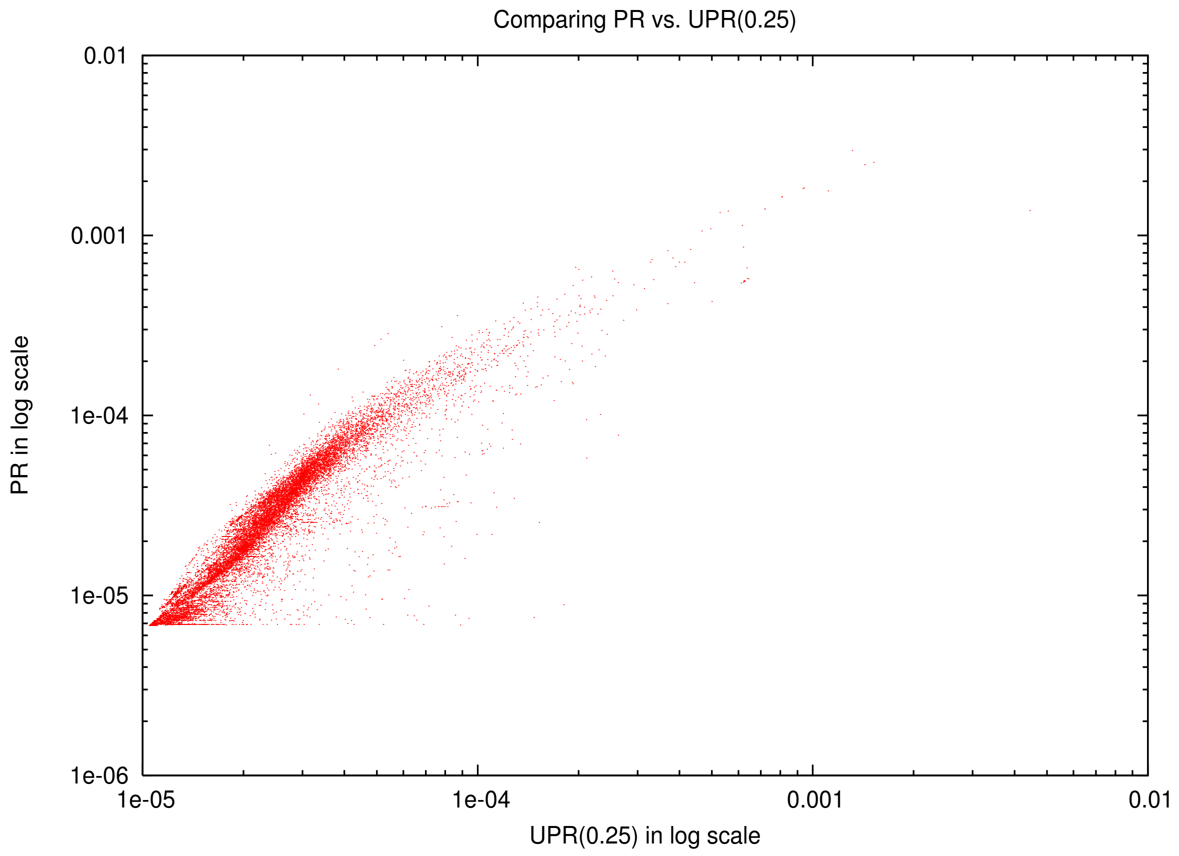


Figure 4.19. Comparing UPR(0.25) and PR scores

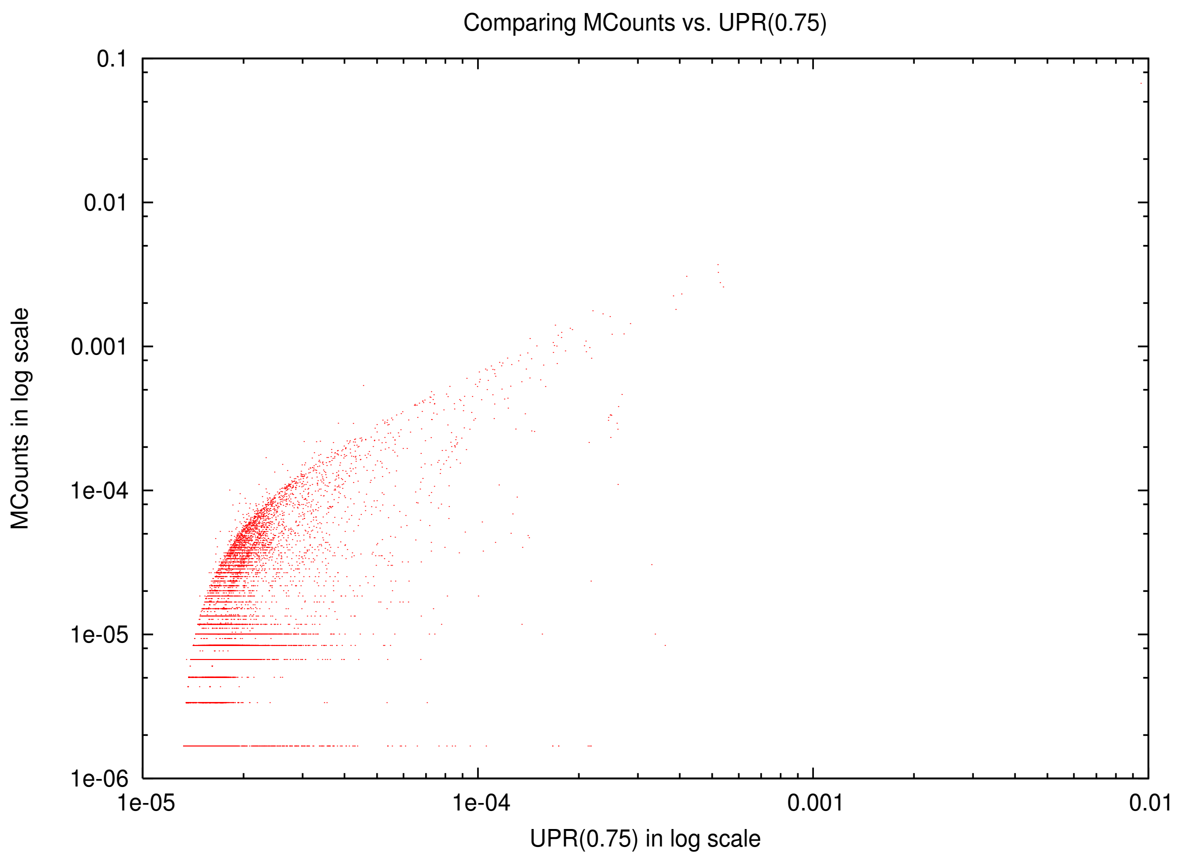


Figure 4.20. Comparing UPR(0.75) and MCounts scores

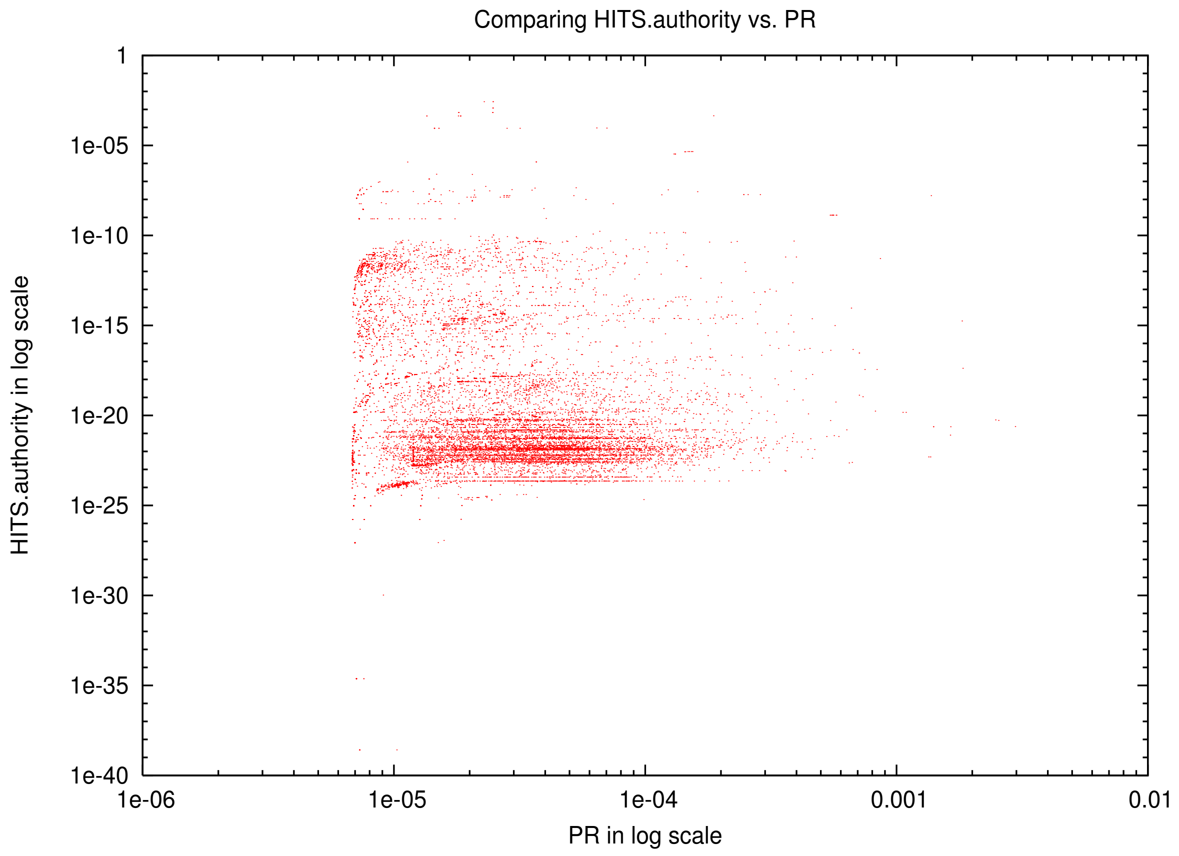


Figure 4.21. Comparing PR and HITS authority scores

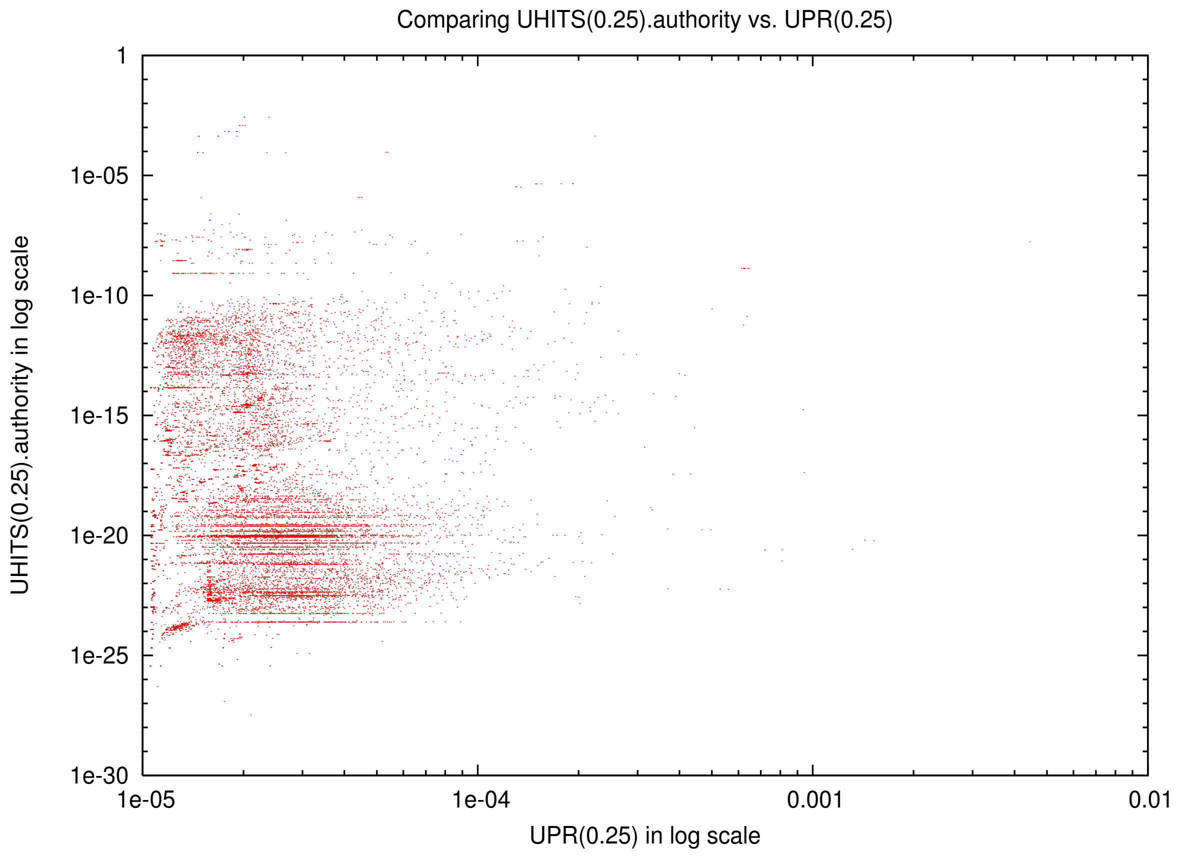


Figure 4.22. Comparing UPR(0.25) and UHITS(0.25) authority scores

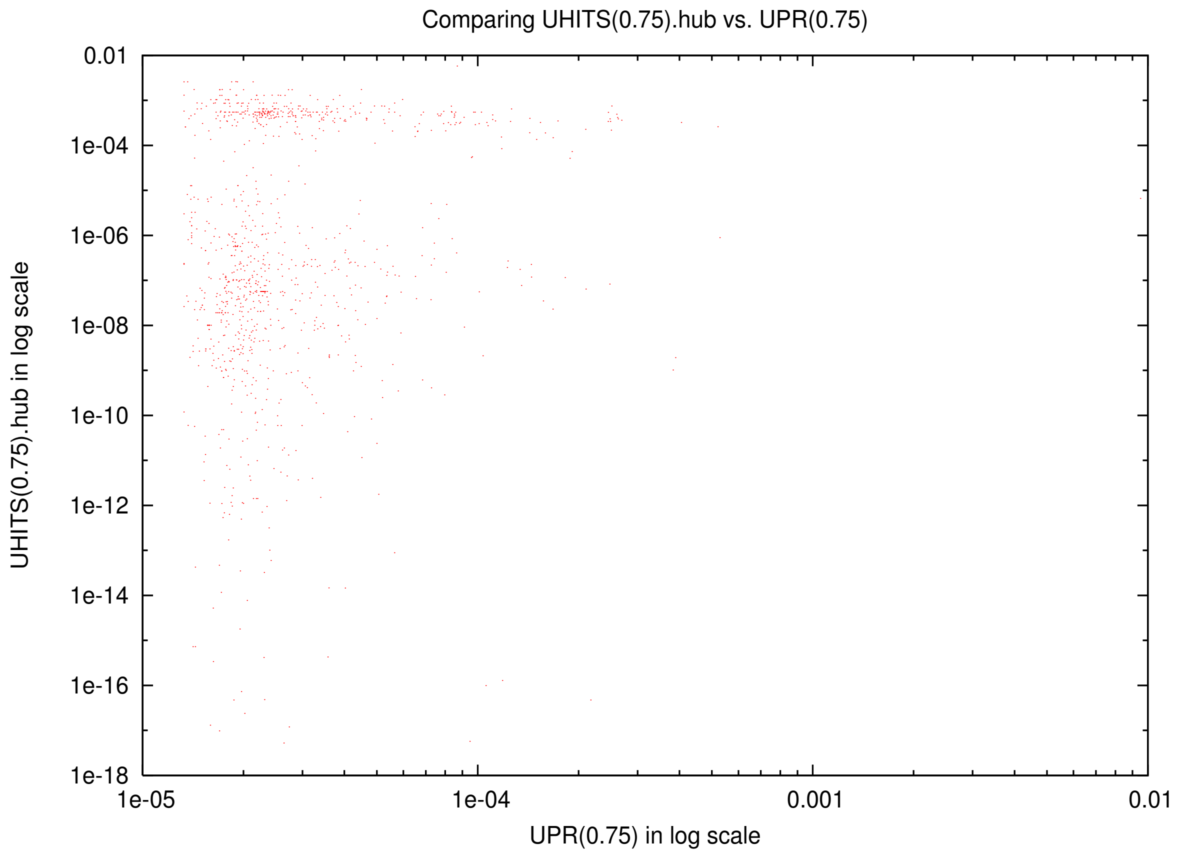


Figure 4.23. Comparing UPR(0.75) and UHITS(0.75) hub scores

as well as vertical line patterns are observed in this figure. This suggests that there were sets of documents for which one method did not offer a distinction, while the other one was able to do so, and vice versa.

The correlation between Counts and MCounts is very high in terms of scores and relatively high in terms of orderings. Figure 4.25 compares the scores of the core 20K documents using Counts and MCounts, while Figure 4.26 focuses on the positions of the documents using these two methods. It is observed that a large portions of the documents are clustered on or around a line, especially when plotted in terms of scores.

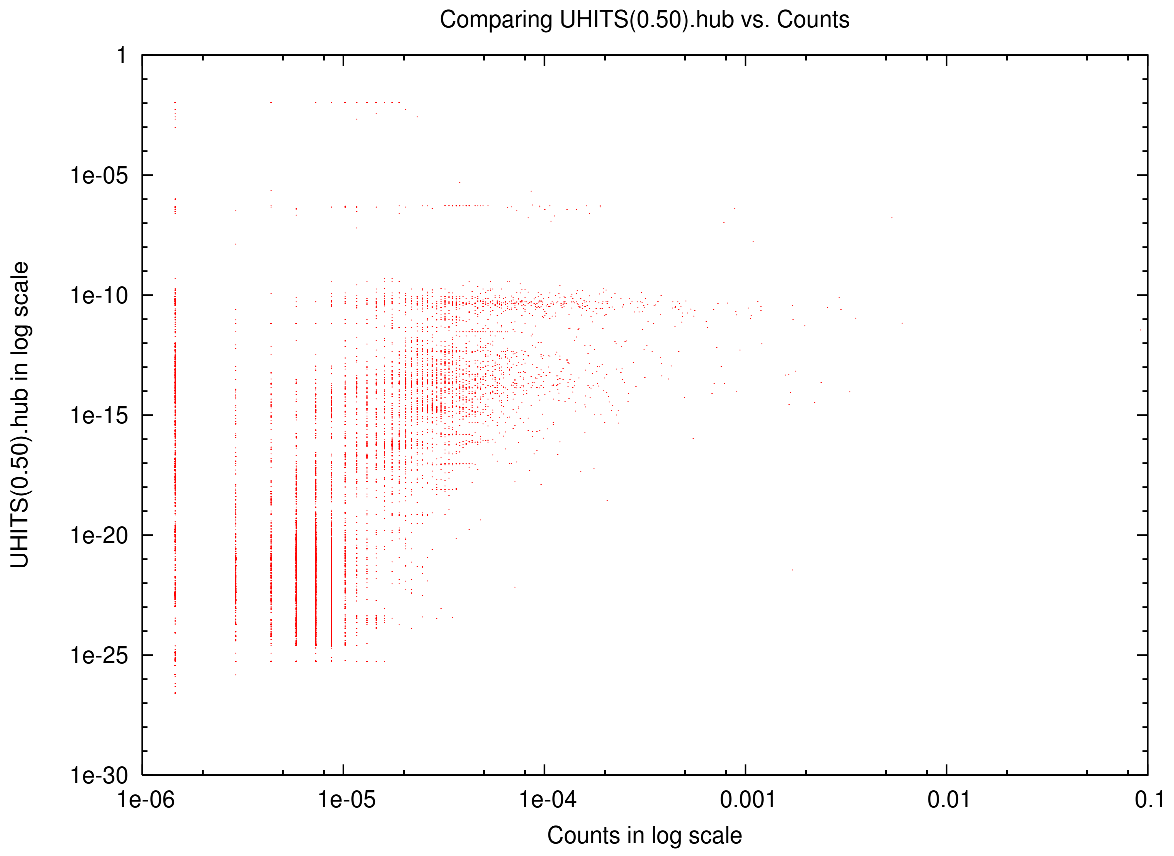


Figure 4.24. Comparing Counts and UHITS(0.50) hub scores

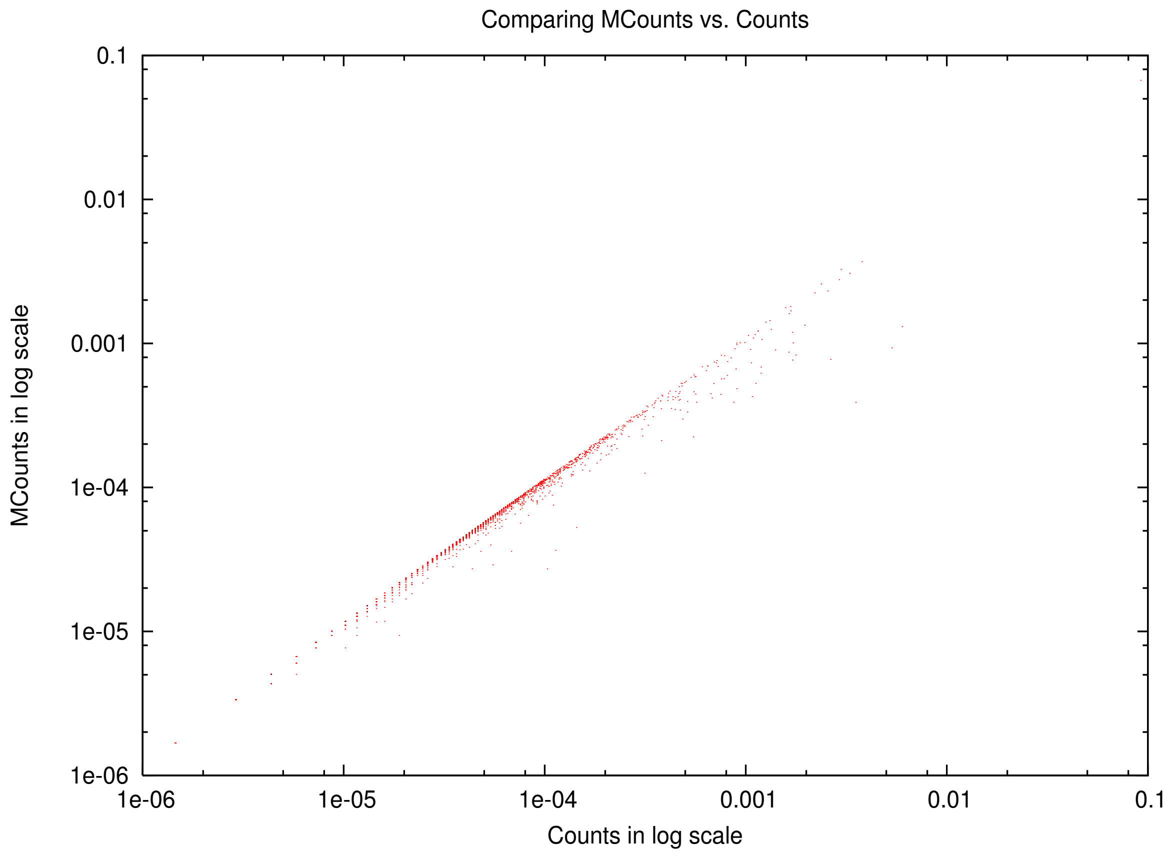


Figure 4.25. Comparing Counts and MCounts scores

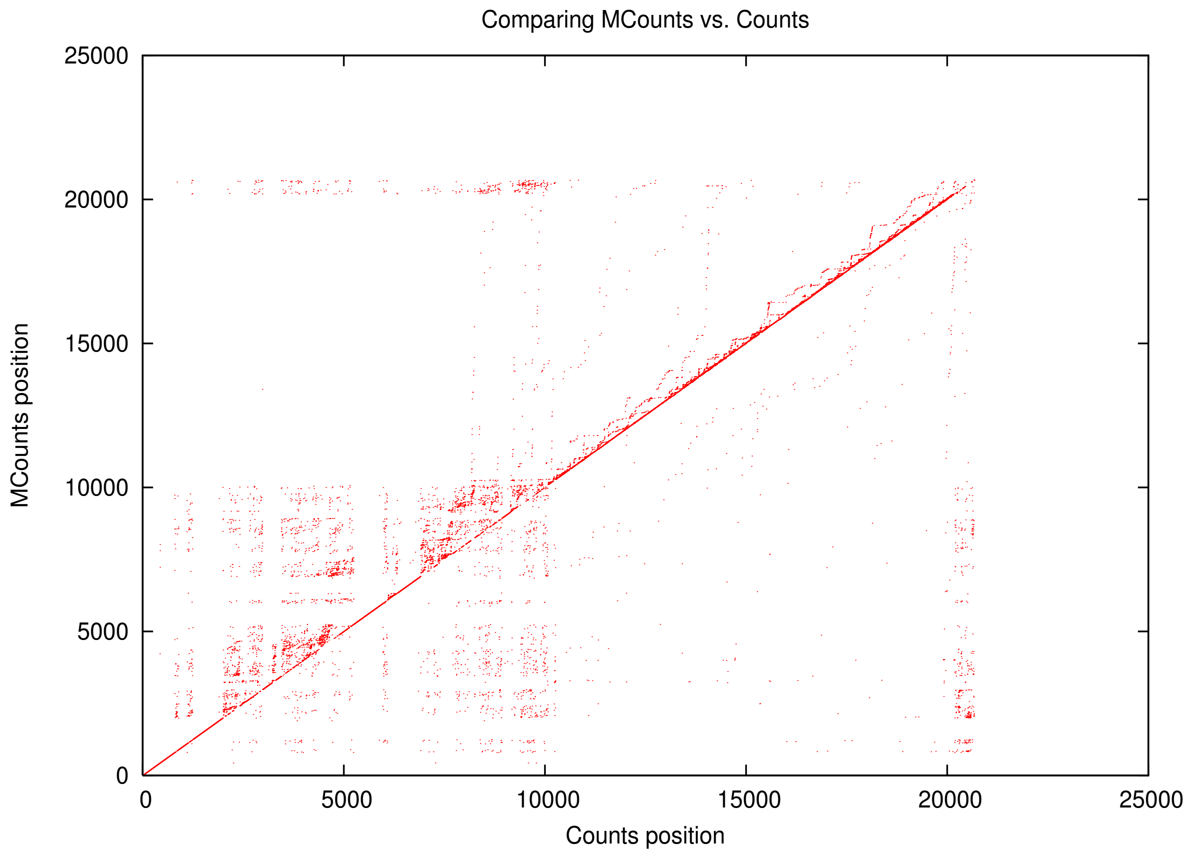


Figure 4.26. Comparing Counts and MCounts ordering

4.4 Query Dependent Comparisons

Explicit as well as implicit relevance feedback have been used in various applications in information retrieval and web search in evaluating the performance of different ranking methods. If explicit user feedback is used, methods are evaluated according to the relevance judgements provided by human experts for a selected subset of queries. This approach has the benefit of offering fairly accurate results for the selected cases. However, due to practical reasons, only a small fraction of the possible cases, queries, and users can be sampled. Evaluation methods based on simulated or implicit relevance information have been proposed as an alternative in the lack of explicit judgements. These approaches make use of statistics/observations that can be collected without user input, either by simulating user judgements via alternative methods/measures [30] (e.g. cosine similarity, frequencies of the query terms within the text etc.) or by inferring them [24, 36] (e.g. using position of clicked documents, time spent reading a document etc.). Using automated implicit relevance feedback approaches, although a larger number of users and query types can be sampled, the evaluations tend to be less accurate per query, requiring a larger user base/evaluation period to offer statistical significance.

USearch's user base is not sufficiently large for automated evaluation approaches using implicit relevance judgements. In this part of the experiments, 3 evaluation sets based on a total of 153 queries are offered: 1) A public evaluation, in which all users are welcome to participate, offering a larger number of samples, but with possibly less accurate evaluation per query. 2) A smaller set of relatively well-formed queries, thoroughly evaluated by at least two users per query, offering more accurate evaluations, but fewer samples. 3) A set of queries independent of user judgements, in which the correct answer(s) can be objectively specified before issuing the query. In this case, the queries are limited to a specific query type (searching for a user's home page by issuing the last name as the query). The author (and the co-authors of the related paper [35]) did not participate in these evaluations.

In order to compare different methods in a query based setting, a special version of USearch is developed, in which the users can provide explicit relevance judgements for a given query. Note that this version does not allow selection of methods, parameters, nor the number of links that will be retrieved. All queries are ranked using cosine similarity only, and top 50 results are presented to the user. As a reminder, USearch uses a reranking approach in which relevant items are first filtered out using cosine similarity to the query, and then reranked using the selected quality measure by combining the query similarity score with the selected quality measure by multiplying the two scores. For instance, if PageRank is the selected quality measure, the final score of a document is obtained by multiplying the query similarity score with the PageRank score. Since similarity to the query was part of all ranking methods, it was a natural choice for a relatively "neutral" method to present the results for evaluation purposes.

The procedure for evaluating a single query is as follows:

- The query is issued
- Top 50 results are returned using plain cosine similarity to the query (no additional quality measures are used at this stage).
- User selects up to 5 relevant documents using the checkboxes next to them, and clicks on the submission button once all the selections are satisfactory. Note that, if the submission button is clicked prematurely or by accident, the user may invalidate the submission and start over.
- The same set of results are reranked using the methods under comparison.
- The new positions of the selected documents are recorded using each method. (A better method places these documents in earlier positions in the list).
- Average positions of the selected documents are calculated for each method. (the smaller the average, the better a given method is).

Evaluation version was announced in department's mailing lists, inviting all graduate students, and some of the faculty to participate. The participants were asked to issue queries they are confident with, and not to assume that the most relevant links were placed in earlier positions. A total of 106 queries are collected for which approximately 3.2 links were selected as relevant on the average. This set will be refered as the "public" evaluation set.

The next set of queries are selected via a small group of 5 users based on their interests and familiarity. In this case, each of the queries were discussed beforehand, and at least two users decided on what they were looking for before issuing the query (e.g. searching for a specific software developed by a research group). Each query results were arbitrated by at least two users, and the selection is done by careful examination of the full list. A total of 22 queries were issued. This set will be refered as the "arbitrated" evaluation set.

Finally, the last set offers evaluations where relevant results could be objectively identified independent of user judgements. 25 faculty and graduate student last names are randomly selected and issued as queries. The "finger" utility is used to identify the usernames that has the selected text as their last name. The queries are issued using all methods. The position of the first occurrence of the URLs of the form /~username/ and /~username/index.html are noted for each user. If multiple users have the same last name, the first page corresponding to each user is selected. The average positions of the selected links using all methods are calculated for this set as well.

	Sim	PR	UPR	MCt	Aut	Hub	UAu	UHb
Public	8.1	10.1	9.0	10.3	12.9	11.8	11.6	9.9
Arbitrated	10.1	7.6	4.8	5.2	6.4	11.3	7.9	7.8
Names	3.0	4.8	3.0	3.1	6.7	10.8	8.0	6.4

Table 4.1. Average position of selected documents for different methods under each evaluation set (smaller is better).

The results of the three experiments are summarized in Table 4.1. The columns in the table from left to right are: Cosine similarity only (no reranking), PageRank, UPR(0.75), MCounts (naïve approach), HITS authorities, HITS hubs, UHITS(0.75) authorities, and UHITS(0.75) hubs. Note that the first column shows the average position of the selected documents using the order in which they are presented to the users (cosine). It is not one of the methods under comparison.

We would like to point out that, due to practical limit on number of samples that can be collected via manual evaluations, the results may not be representative of majority of users or queries. Note also that, types of queries that are selected for each of the evaluation sets tend to be quite different. In the arbitrated set, queries were somewhat specific and well-formed, the correct answers were mostly selected before issuing the query, and all results were carefully examined by at least two of the participants. Evaluations for these queries are expected to be more accurate, and relatively free of bias that may be associated with the order of presentation. The queries that are issued on the public set, on the other hand, tend to be more general. Moreover, due to public nature of the evaluations, the selection process should not be expected to be as thorough as the arbitrated set. Possible bias due to order of presentation should also be investigated. Note that the last set (name search) does not depend on user judgements nor on the order of presentation (bias is not applicable).

We observed that, in 12 of the queries issued during the public evaluation, the selected documents were within the top 3 results presented using cosine similarity. Upon closer examination of these queries, for a subset of them, we did not see an apparent reason why one of the subsequent links was not considered as relevant. One possible explanation is that, if two or more links are subjectively equally relevant, the user might have chosen the link that appeared first. It may also be the case that, once a sufficient number of relevant links have been identified, some of the users may not have examined the rest of the list in similar detail. If we remove a portion of the suspected queries, average position of the cosine in the public set becomes slightly higher then UPR (in favor of UPR), but still lower than the rest of the methods. However, we chose to rely on the subjective judgements of the users, and not to modify/filter the public set unless the user explicitly invalidated a submission (usually when the submission button is hit by accident).

The fact that the average position of cosine similarity being smaller than the rest of the methods in the public set may suggest that at least some of the users could have been biased by the order of presentation (towards cosine). This could suggest that, for this set, if the correlation between cosine ordering and the ordering of a particular method is higher than the rest of the methods, this particular method could have been favored by the evaluation process. We calculated the average spearman correlation of the ordering using cosine vs. every other method for this particular set. The average correlation was highest between Sim and UHITS hub (0.56), and ranged from 0.22 to 0.34 for the remaining methods except HITS authority (0.08). Reissuing sampled queries from the public set using regular USearch, we noticed that, for a portion of the queries, UHITS hub was not able to distinguish between URLs after top few documents. Thus cosine order was preserved in the remaining portion of the list. This behavior was also observed using UHITS authority, but to a lesser degree. Effectively, for some queries, UHITS hub and UHITS authority pulled up a few highly connected/used URLs in top positions and kept the cosine order in the rest. If these were the URLs that happened to be selected by the user (possible if the query searches for popular pages), then these two methods performed relatively well. However, for other queries, they did not perform significantly better than pure cosine. For PR, UPR, and MCounts, a similar behavior was not observed, especially PR and UPR were able to offer a global ordering among the query results. Note that these observations are compatible with the analysis in Section 4.2.

In all three sets, UPR performed better than the rest of the methods under comparison, while PR and MCounts performed relatively well. UHITS hub's and UHITS autority's performance cannot be reliably judged using the public set, as they failed to offer a distinction between pages, and may have been favored by possible bias towards cosine for this particular set. A user independent observation that follows from the results is that, UPR and MCounts place user home pages in relatively top ranks compared to other quality measures. This is compatible with the early observation that these pages are, in general, heavily used. Another observation is that, UPR, MCounts, and PR seem to perform better than the rest of the quality measures, for the queries and the users sampled. They also perform better than the HITS versions modified via usage except in the case of UHITS hub in the public set.

Due to the nature of name search, it is expected that the cosine order produces reliable results (users home pages are very likely to contain their last name in higher frequencies compared to a random page). In this evaluation set, UPR performed as good as cosine ordering, while MCounts, performed similarly, with a slightly higher average. In the public set, cosine similarity is better than any of the methods, while UPR is slightly higher. However, there is reasonable indiction that, for at least 10% of the queries, the users might have been biased by the presentation order. In the arbitrated set, in which presentation bias is less likely to occur, UPR (and MCounts) perform significantly better than other methods (including cosine ordering). Note that, for this set, all methods, except HITS Hub, perform better than cosine as well.

4.5 Summary

In this chapter, all methods are first examined in terms of distinguishing power they offer, and how smooth they behave under various parameter values. Extensive global and pairwise comparisons showed that PageRank variants are stable, offering distinguishing power between pages, and providing a smooth global ordering. It is also verified that at one end of the possible values for the parameters a_1 and a_2, UPR behaves similar to PageRank, and on the other end, it gets more and more correlated to pure usage based naïve methods. Counts/MCounts also offer decent distinguishing power, but they are not as smooth as PageRank variants. HITS variants were the worst methods among the ones compared, failing to offer distinguishing power for a large subset of pages (in some cases, converging to zero for more than half of the data set). Also, their behavior under different parameter values were not as stable and smooth as other methods.

A representative of each method is then compared against others in a query dependent setting using explicit user judgements in two evaluation sets: 1) a public evaluation open to all participants, offering larger number of samples, but potentially less accurate results, also being more prone to bias. 2) An arbitrated set, offering a thorough evaluation by at least two users for a smaller set of queries, less prone to bias, but offering fewer samples. A third set for a specific query type (name search) is also provided, in which the relevant documents can be identified independent of user judgements. Results showed that UPR performed better than all methods, while MCounts and PageRank performed better than the remaining in most cases.

Chapter 5

Discussions and Future Directions

*.cs.umn.edu domain is reasonably sized for a departmental home page/site, but it is easily dwarfed compared to large corporate intranets. It may be very interesting to test the proposed methods or others on a larger scale or on different domains, where some of the issues that do not appear in our data set may surface (e.g. partial usage information, uneven sampling, and significantly higher usage based voluntary/involuntary perturbations and spamming). However, obtaining usage information from an organization has serious security and privacy issues that would not be easily addressed in an academic research environment. Organizations may perform a similar study themselves and publish a summary of the results, but in this case, it would be significantly more difficult to reproduce the findings. Future research in the area of usage statistics in search is likely to face the above challenges.

5.1 Quality Signals and Search

Almost all desirable quality signals in search have a good benefit to cost ratio. The benefit can be measured in a number of ways in terms of the improvements in ranking quality, and the cost can be measured in terms of number of cases in which the signal actually harms the ranking quality, and by how much. Although there is not a widely used benchmark in the literature, one may easily develop a number of metrics reflecting the above, examining the signals from different perspectives. A good quality signal should ideally offer high benefit at no cost. However, this is practically never the case in real life. In general, aggressiveness of a signal and the parameters controlling/combining signals are adjusted in a way that the benefit to cost ratio is kept within a reasonable range.

It is also important to note that recent search engines potentially employ/combine a large number of quality signals. The benefit of each signal should be evaluated separately as well as in conjunction with other signals. Given two signals may be quite strong independently, but, when combined together with other signals, may not improve the quality significantly. For example, they may be highly correlated, and once one is included, the other may not offer a significant additional benefit. However, this does not suggest that one of the signals (e.g. the one that is added later on while the other is already being used) is inferior. In general, search engines should have a pool of signals that are continuously evaluated. Depending on the dynamics, a signal that is left out as redundant may become important in the future, or may replace one that became weaker. The best combination strategies may also change in time.

The signals examined in this study were selected among the ones that seemed likely to be primary/core signals for the domain in focus. These are:

- Cosine similarity, a simple but effective information retrieval relevance score.
- Usage statistics, a popularity signal reflecting the preferences/perspective of the users.
- Link analysis, which can be considered as a popularity signal from the point of view of page authors (Although this distinction may be blurred with increased use of Blogger [5], WikiWeb [52], and other facilities making it practical for the users to modify or append to online pages).

Other signals improving or complementing some of the above, as well as more orthogonal signals are possible. Some of these signals may also be domain specific. This study primarily examined the intelligent combination of two of the core signals, specifically, merging link analysis with usage statistics into one signal by a reasonably complex approach. The resultant signal (UPR or UHITS) is then combined with cosine similarity using a simple approach. This is, by no means, an exhaustive exploration. The method selection and various choices were informed ones, limiting the scope of the analysis to a manageable size, aiming to start by exploring the baseline and the methods that showed more promise. One possible alternative that was not examined in this study is to treat all three signals independently first, and then combine them. This approach, and various other possibilities for combining the above signals were out of the scope of this study, and may be worth considering for future research.

5.2 Usage Signals and Web Search

Although, the focus in this study was the applicability on a site specific search setting, UPR and similar usage based or augmented signals can also be applied on a global scale if usage statistics can be collected via specialized browsers/tools such as Google Toolbar, or via caching proxies. Usage statistics would not be complete (only a small portion of users can potentially be sampled using current approaches), and the sampling process may introduce additional bias. For example, users of a particular search engine and its tools may be less likely to visit pages in competitors; or computer and security aware users may be more likely to minimize their online presence, choosing a higher level of privacy, hence end up being under-represented. Nevertheless, usage information is available to a number of search engines, and it can potentially be an important signal for Web scale search as well.

In terms of computation in large scale, the process required for incorporation of usage statistics in one way of another is not significantly different than other signals already used in modern search engines. For example, without going into specific details, implementing UPR on a large scale should not be a technical challenge for a leading search engine company such as Google, which already has significant infrastructure that can be used to facilitate distributed/parallel computation such as [11, 14].

5.2.1 Usage Based Spamming

During the early days of Web search engines, one could boost the score of a page by modifying the page alone (adding/removing keywords, modifying the frequencies etc.). In recent search engines using link analysis, in order to boost the score of a page considerably, one needs to boost the link structure around that page, requiring modification of a number of pages as opposed to a single page. Hence, link analysis is historically considered to offer relative resistance against spamming compared to the early, simple approaches. If usage augmented link analysis algorithms are used, not only pages/links must be created/modified around the page that one desires to boost, these pages/links should also have relatively high usage, offering an additional signal that can be used against spamming. The other side of the coin is that, by introducing a new type of information (link analysis), a new door that can be abused has been opened (link based spamming). Similarly, adding usage information among the signals opens up a new, potential way of perturbing the results (usage based spamming).

UPR formulation offers relative resistance against all types of usage based spamming that are investigated in this study (unlike previous attempts [56, 33], which offered none). An improvement is also proposed, deemphasizing pages and links that are visited by few users many times, and emphasizing the ones that are used by various users. This improvement does not change the overall rankings of majority of the pages, but helps in filtering out a small subset of pages that are accessed by very few people many times. When UPR is applied with the proposed improvement and appropriate parameters, in order to boost the score of a page significantly, not only one needs to build a link structure around that page (a "link farm"), but also, the pages/links need to be supported via a sustained traffic received from a number of distinct sources.

In Web search, usage based spam (or click spam), is becoming an increasing problem, especially when advertisement and commercial queries are involved. Various heuristics or machine learning techniques can be used to identify usage based spam, or to reduce its effects. The approach used in this study (log transform of counts, deemphasizing the contribution of a single entity in a given time window) is a reasonable approach for the domain in focus. One may also imagine various alternatives that may be equally viable; or ones that are better suited for other domains. As an example. if excessive spam is expected, more aggressive approaches may offer a better solution (e.g. a binary: contribution of 0 or 1 per entity per time window). It is also possible to treat a given IP block or set of IPs as a single entity. Machine learning approaches used for detecting automated agents/robots (e.g. [45]), may also be adapted to this domain.

Spamming will potentially continue to be a problem in search when significant commercial benefits/losses are involved. For any given algorithm or signal, it should be possible to devise a scheme to exploit it somehow. One of the reasonable strategies search engines may employ could involve effectively increasing the cost of spamming, hence reducing the incentive for it. One of the questions with regards to usage statistics in Web search is that, is it possible to find a set of usage spam prevention techniques/algorithms that will increase the effective cost of the overall exploits? Or will it make it actually easier than existing spamming approaches? It is reasonable to argue that usage augmented link analysis techniques could increase the cost of link based spamming (since not only a link farm needs to be build, but also, pages/links need to be supported by a sustained usage from multiple sources). However, it may very well be the case that usage based spamming may actually be cheaper than link based spamming. All of these factors affect the benefit to cost ratio of usage based signals, and need to be taken into consideration while determining to what extend usage information can be effectively used in Web search.

Spamming is not a real issue within intranets. Changes in benefit to cost ratio for this signal due to spamming is not as important as it can potentially be in Web search. Intranet search engines may readily incorporate usage information with minimal spam/perturbation resistance, and potentially add a strong signal for improving ranking quality.

Chapter 6

Concluding Remarks

A number of usage based ranking approaches (UPR, UHITS, and Counts/MCounts) are introduced in this study. These approaches make use of different types of usage and static linkage information. UPR and regular PageRank are compared against each other in detail, providing intuition about the types of pages that are favored by these two methods, and examining the effects of adding increasing amount of usage information. All usage based methods are also compared against each other and against classical approaches (PageRank and HITS). The behavior of different methods, their ability to distinguish between pages, how smooth/sharp the effects of a given parameter is, and for which parameter values a given method is more/less correlated with other methods are examined in detail. A representative of each method are also compared in a query based setting using blind evaluations, as well as using a name based search setting, independent of user judgements.

6.1 Usage Information in Site Specific/Intranet Search

Intranets tend to have relatively poorer connectivity compared to the Web, potentially reducing the effectiveness of link analysis approaches that worked reasonably well for Web search. This study explored the possibility to use usage statistics as an additional signal in intranet/site specific search domain. A number of usage based approaches are introduced and implemented. Two of these methods modify and augment popular link analysis approaches that are already examined in the literature extensively (PageRank and HITS). Two naïve methods, Counts and MCounts, are also implemented, providing a pure usage based baseline benchmark to compare against. The naïve methods are similar in spirit to the ranking approach used in an early commercial search engine, DirectHit, but employ complete usage statistics as opposed to being limited to search result clicks. The proposed methods are also selected to offer reasonable diversity in terms of the types of information they use: UPR uses page visit statistics, as well as link traversal statistics (combined with static linkage structure); UHITS uses link traversal statistics (and static linkage information) only; and Counts/MCounts are purely based on usage, employing page visit statistics only.

Compared to Web search engines which have a limited view in terms of usage statistics (through specialized tools, browsers, proxies etc.), the owner of a site/intranet can potentially have access to extensive usage logs. This may not be practically true for all intranets today due to distributed servers and potential lack of coordination within the organization. However, the data is there, and it belongs to the organization. If there is a motivation, processing the logs is not a technical challenge. For example, each server (or cluster of servers) responsible for a unique portion of the intranet may process its view of the usage graph and send the compressed information to a central location that carries out the ranking computations. Note that the graphs can be combined without loss of information under certain conditions. For example, if multiple servers are responsible for a given resource, one possible approach is to aggregate their logs for that resource before applying time window analysis if non linear functions such as log transform of counts are used. After this step, the entry for that resource can be readily merged with other resources.

6.1.1 UPR, a Promising Signal for Intranet Search

Experimental results suggest that among the methods that are examined, UPR performs consistently better than other methods, has a number of desirable properties, improves the ranking quality for a large number of cases over existing signals, and did not practically hurt the quality under any circumstance (even when compared against cosine similarity, which may have a presentation bias in one experiment (public evaluation set), and is favored in another (name search) due to the specific query type). UPR, a combined signal using both link analysis and usage statistics, shows promising potential as a well-behaving signal for this domain, offering significant quality improvements over a set of query types, with almost no cost at others.

UPR has a number of properties that previous attempts of incorporating usage information into link analysis [56,33] failed to address: Ability to control usage emphasis smoothly, relative resistance against usage based spamming, scalability, offering a global ordering, and ability to work with limited usage gracefully. The formulation inherits basic PageRank properties. It is intuitive, general, and flexible. Usage emphasis can be controlled smoothly via the two UPR parameters. Experiments suggest that, unlike UHITS, UPR's behavior can be adjusted smoothly from pure static structure based to pure usage based ranking.

The formulation is also quite inexpensive and scalable. Usage graph can be updated incrementally and efficiently. UPR iterations are only a small constant times more expensive than PageRank iterations in terms of speed and storage. An optimization is also proposed (Eq. 2.12) that further speeds up the iterations, making them comparable to PageRank iterations.

UPR can be used in the presence of partial information. Although scores of pages for which limited usage statistics are available can be lower than what they would be in the presence of full usage statistics, if these pages are pointed to by highly ranked pages, their scores will still be relatively high. Unlike methods that apply usage statistics to boost the scores per page basis (e.g. counting number of hits to a page and using it as a quality measure), UPR scores of pages are gradually affected as less and less usage statistics are available, converging to regular PR at the extreme case.

Unlike Web search, an important property of intranets is that, in general, there is little or no incentive for spamming the results within a single organization. In this particular domain, usage signals and usage augmented link analysis, even employing modest spam/perturbation prevention techniques, may easily offer a reliable signal with a desirable benefit to cost ratio. Usage signals may become one of the strongest signals for enterprise search in the future.

Bibliography

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In J. B. Bocca, M. Jarke, and C. Zaniolo, editors, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, 12–15 1994.

[2] F. Anklesaria, M. McCahill, P. Lindner, D. Johnson, D. Torrey, and B. Alberti. The internet gopher protocol (a distributed document search and retrieval protocol, rfc 1436). 1993.

[3] T. Berners-Lee, R. Fielding, U. Irvine, and L. Masinter. Uniform Resource Identifiers (URI): Generic Syntax, RFC2396. 1998.

[4] K. Bharat and M. R. 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.

[5] Blogger. http://www.blogger.com/.

[6] A. Borodin, G. O. Roberts, J. S. Rosenthal, and P. Tsaparas. Finding authorities and hubs from link structures on the world wide web. In *World Wide Web*, pages 415–429, 2001.

[7] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[8] K. G. Coffman and A. M. Odlyzko. Growth of the internet. *Optical Fiber Telecommunications IV B: Systems and Impairments*, pages 17–56, 2002.

[9] cURL. http://curl.haxx.se/.

[10] M. Cutler, Y. Shih, and W. Meng. Using the structure of HTML documents to improve retrieval. *USENIX symposium on Internet Technologies and Systems (NISTS'97)*, pages 241–251, 1997.

[11] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Fransico, CA, December 2004.

[12] Directhit. http://www.directhit.com/.

[13] R. Fagin, R. Kumar, K. S. McCurley, J. Novak, D. Sivakumar, J. A. Tomlin, and D. P. Williamson. Searching the Workplace Web. In *WWW12 Conference Proceedings", May 20–24, 2003, Hungary*, 2003.

[14] S. Ghemawat, H. Gobioff, and S.-T. Leung. The google file system. 19th ACM Symposium on Operating Systems Principles, Lake George, NY, October 2003.

[15] D. Gibson, J. M. Kleinberg, and P. Raghavan. Inferring Web Communities from Link Topology. In *Proceedings of the 9th ACM Conference on Hypertext and Hypermedia*, pages 225–234, Pittsburgh, Pennsylvania, June 1998.

[16] Google. http://www.google.com/.

[17] J. Han and M. Kamber. *Data Mining: concepts and techniques*, pages 437–439. Morgan Kaufmann, 2001.

[18] D. Harman and G. Candela. Retrieving records from a gigabyte of text on a minicomputer using statistical ranking. *American Society for Information Science*, 41(8):581–589, 1990.

[19] T. Haveliwala. Topic-sensitive pagerank. In *Proceedings of the Eleventh International World Wide Web Conference, Honolulu, Hawaii, May 2002.*

[20] T. Haveliwala. Efficient computation of pagerank. Technical Report 1999-31, Stanford Digital Library Technologies Project, 1999.

[21] HTTrack. http://www.httrack.com/.

[22] G. Jeh and J. Widom. Scaling Personalized Web Search. In *WWW12 Conference Proceedings", May 20–24, 2003, Hungary*, 2003.

[23] S. D. Kamvar, T. H. Haveliwala, and C. D. Manning. Extrapolation Methods for Accelerating PageRank Computations. In *WWW12 Conference Proceedings, May 20–24, 2003, Hungary*, 2003.

[24] J. Kim, D. Oard, and K. Romanik. Using implicit feedback for user modeling in internet and intranet searching, 2000.

[25] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[26] L. Kleinrock. Information flow in large communications networks. *RLE Quartely Progress Report*, July 1961.

[27] M. Koster. A Standard for Robot Exclusion, http://www.robotstxt.org/wc/norobots.html. 1999.

[28] S. Kumar, B. U. Oztekin, L. Ertoz, S. Singhal, E.-H. Han, and V. Kumar. Personalized profile based search interface with ranked and clustered display. In *2001 International Conference on Intelligent Agents Web Technologies and Internet Commerce - IAWTIC'2001.*

[29] R. Lempel and S. Moran. The Stochastic Approach for Link-Structure Analysis (SALSA) and the TKC Effect. In *WWW9 Conference Proceedings", May 15 - 19, 2000, Amsterdam*, 2000.

[30] L. Li and L. Shang. Statistical performance evaluation of search engines. In *WWW10 conference posters, May 2–5, 2001, Hong Kong.*

[31] C. D. Manning and H. Schtze. *Foundations of Statistical Natural Language Processing*, chapter 5. MIT Press, 1999.

[32] MEARF. http://mearf.cs.umn.edu/.

[33] J. Miller, G. Rae, and F. Schaefer. Modifications of Kleinberg's HITS Algorithm Using Matrix Exponentiation and WebLog Records. In *ACM SIGIR Conference posters, September 9-13, 2001, New Orleans, Louisiana, USA*, pages 444–445.

[34] G. Newby. Information space based on HTML structure. In *TREC9*, 2000.

[35] B. U. Oztekin, L. Ertoz, V. Kumar, and J. Srivastava. Usage Aware PageRank. In *WWW12 conference posters", May 20–24, 2003, Hungary*, 2003.

[36] B. U. Oztekin, G. Karypis, and V. Kumar. Expert agreement and content based reranking in a meta search environment using Mearf. In *proceedings of the Eleventh International World Wide Web Conference, May 7–11, 2002, Honolulu, Hawaii.*

[37] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[38] Pavuk. http://www.idata.sk/ ∼ondrej/pavuk/.

[39] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[40] W. Recommendation. HTML 4.01 Specification, http://www.w3.org/TR/html4/. 1999.

[41] M. Richardson and P. Domingos. The Intelligent Surfer: Probabilistic Combination of Link and Content Information in PageRank. In *Advances in Neural Information Processing Systems 14*, 2002.

[42] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[43] A. Schapira. Collaboratively searching the web – an initial study. Master's thesis, 1999.

[44] SWISH-E. http://swish-e.org/.

[45] P.-N. Tan and V. Kumar. Discovery of web robot sessions based on their navigational patterns. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, volume 6, pages 9–35, 2002.

[46] A. S. Tanenbaum. *Computer Networks, Third Edition.* Prentice-Hall, 1996.

[47] Teoma. http://www.teoma.com/.

[48] USearch. http://usearch.cs.umn.edu/.

[49] C. van Rijsbergen. *Information Retrieval, Second Edition.* Butterworths, 1979.

[50] W3C. Cascading style sheets, http://www.w3.org/Style/CSS/. 1999.

[51] Wget. http://www.gnu.org/software/wget/wget.html.

[52] Wikiweb. http://www.wikiweb.com/.

[53] R. H. Zakon. Hobbes' internet timeline, http://www.zakon.org/robert/internet/timeline/.

[54] B.-T. Zhang and Y.-W. Seo. Personalized web-document filtering using reinforcement learning. *Applied Artificial Intelligence*, 15(7):665–685, 2001.

[55] M. Zhang, R. Song, and S. Ma. DF or IDF? On the Use of HTML Primary Feature Fields for Web IR. In *WWW12 conference posters, May 20–24, 2003, Budapest, Hungary.*

[56] J. Zhu, J. Hong, and J. G. Hughes. Pagerate: counting web users' votes. In *Proceedings of the 12th ACM Conference on Hypertext and Hypermedia*, pages 131–132, Arhus, Denmark, 2001.