Imperial College of Science, Technology and Medicine
(University of London)
Department of Computing

Distributed Asynchronous Link-Based and Content-Sensitive Pagerank Algorithms

Saar Miron
sm1603@doc.ic.ac.uk

Project supervisor: Dr. Jeremy Bradley
Second marker: Dr. William J. Knottenbelt

September, 2004

Submitted in partial fulfillment of the requirements of the MSc Degree in Computing Science of the University of London and the Diploma of Imperial College of Science, Technology and Medicine

Abstract

This paper presents a fully distributed implementation of a non-iterative algorithm for computing Pagerank in large scale hypertext environments. The implementation introduced here does not imitate the centralized Pageank algorithm implemented effectively in Google [Goo], or convert it into a distributed system. Instead, it offers a different kind of implementation, one that is suitable for both the scale and the dynamic behavior of the Web. In view of this, it is possible that the distributed ranking implementation presented here may be a candidate for the replacement of the traditional centralized Pagerank computation.

This paper criticizes the use of pages' link-structures as the sole criterion for determining the "importance" of pages in a ranking algorithm. Instead, it argues that the weight of links in a ranking equation should be partly determined by the content relevance of the connected pages. Content analysis can be effectively incorporated into a distributed Pagerank algorithm using the fact that nodes (Web-servers) have immediate access to their pages' contents. This paper proposes incorporating content analysis into a Pagerank algorithm in order to improve both rank quality and convergence performance.

The project source code together with all documentations and final report can be found at:

www.doc.ic.ac.uk/∼sm1603/project/

Acknowledgements

Firstly, I would like to thank my supervisor, Dr. Jeremy Bradley, for his enthusiastic approach, valuable advice and help throughout the project.

I would also like to thank Dr. William J. Knottenbelt for agreeing to be my second marker.

I am indebted to Larry Booker for the English corrections which improved immensely the quality of this paper.

A big thank to my friends Jean-Philippe Sayler, Riccardo Ussani, Daniel Boswell, Luke Dickens, Douglas De-Jager, Lida Tsimonou, Claudio Jolowicz, Nicola Ciancaglini and Matthew Cook for sharing ideas and keeping me smiling for long working hours in the Lab.

Special thanks to Dana Margalith, my loved girlfriend, for her sound advice and for suffering me during the last weeks of the project.

Finally, I wish to express my heartfelt gratitude to my parents Aviva and Eliezer Miron, to my sisters Ella and Zohar, and to Yaron Spector. Without your support this would not have been possible for me.

Contents

List of Figures

0.1 Glossary

PR - Pagerank algorithm

QD-Pagerank - Query-Directed Pagerank algorithm

TFIDF - Term Frequency Inverse Document Frequency

Web-graph - a link-structure representation of a collection of pages

Term - sequences of non-space characters found by filtering and normalizing an HTML page

Term-vector - a sequence of term-weight pairs found in a page

Term-index - a sequence of term-frequency pairs taken from a large collection of pages

Term-lexicon - a sequence of significant terms (used for filtering term-vectors)

Outbound-links - hypertext links contained in a page that point to other pages (also called outlinks)

Inbound-links - hypertext links in other pages that link to a page (also called inlinks and backlinks)

SEO - Search Engine Optimizations

ODP - Open Directory Project

Link-based Pagerank - Pagerank calculated by algorithm which rely solely on the link structure of the Web-graph

Content-match - the inner-product of two term-vectors, normally represented by the percentage of matching terms

Chapter 1

Introduction

1.1 Overview

Introduced in 1998, Google and the Pagerank algorithm resulted from the project of two PhD students, S. Brin and L. Page. Pagerank was presented as a method for ranking Web pages objectively by measuring “he human interest and attention devoted to them” [PB98]. The algorithm measures the global ”importance” of pages using the link-structure of the Web. It is implemented in a centralized manner in Google [Goo] and has become an important ranking mechanism in search engines.

The intuition underlying the Pagerank approach is that in a hypertext environment pages with many inlinks are more likely to be of high quality than pages with few inlinks, given that the author of a page will normally include a number of links to other pages he or she believes contain related content.

As the size of the Web grows, the centralized computation of Pagerank becomes a major weakness. In 1998, Page and Brin estimated that the Web contained some 150 million Web pages [PB98]. Today Google’s database contains more than 4 billion pages [Goo], representing less than 50% of the total estimated number of Web pages [SEO].

The scalability problem, and the dynamic character of many Web pages (which frequently change their content and their link structures), have become major problems of centralized search engines. These challenges, together with the absence of a good ranking mechanism for P2P networks, have led researchers to look for alternative methods of ranking pages.

The research undertaken in this project was motivated by the shortcomings of the centralized Pagerank algorithm implementation. The project argues that the characteristics of the Web cannot be reflected correctly by centralized search

engines. It proposes solutions to the scalability and availability problems as well as ways to improve on the ranking quality of the centralized Pagerank algorithm.

The project introduces two distributed algorithms for computing Pagerank. The first, a distributed asynchronous non-iterative algorithm, is based on the link-structure of pages maintained in Web-servers. The second, a content-sensitive Pagerank algorithm, combines content analysis of pages with a distributed Pagerank computation.

1.2 Distributed Pagerank Algorithms

Designing a distributed Pagerank algorithm is not a trivial task. The implementation of a distributed system for solving large scale computation problems requires careful examination of the behavior of the network used.

A straightforward approach to designing a distributed Pagerank algorithm is simply to scale the centralized algorithm to a distributed environment. This approach is problematic, however, as the Pagerank algorithm is computed iteratively: every iteration step in the computation depends on the results of the previous iteration. Thus a synchronized method is needed. Furthermore, applying such a method in a large scale distributed system may be impractical due to the nature of a network. Delays, failures and congestions, which characterize any large scale networks, conflict with the requirement for synchronization of iteration.

This paper presents a distributed asynchronous non-iterative algorithm for computing Pagerank in large scale hypertext environments. It criticizes the use of synchronization in distributed computations, as put forward in various papers (such as [SSW03] and [KSB03]). By contrast, the approach introduced here is not willing to imitate the centralized Pagerank algorithm or to convert it into a distributed system. Instead, there is an attempt to offer a different kind of implementation, one suitable to both the scale and the dynamic behavior of the Web.

The ranking system developed in this project seeks to exploit the natural partition of the Web. By this is meant the immediate access of web-servers to their pages and the fact that most of the links in the Web connect pages that belong to the same web-server.

In later chapters various aspects of the algorithm are discussed in detail and solutions for the major problems raised are suggested.

1.3 Content-Sensitive Pagerank Algorithms

This paper criticizes the use of pages' link-structures as the sole criterion for determining the "importance" of pages in a ranking algorithm. It claims that link-based Pagerank algorithms may only rank pages by their popularity, not necessaraly their quality.

The implementation of the Pagerank algorithm in Google and other search engines appears to have encouraged the development of a new industry, Search-Engine-Optimizations [SEO], that specializes in manipulating web-sites' structures in order to acheive higher ranking for pages. Simple techniques of interlinking pages for the purpose of improving Pageranks are common knowledge these days. Commercial companies guarantee "Top Ten" search results in Google [Top] for any key-word chosen. As a consequence, a page's ranking is arguably often more dependant on a programmer's skills than on the page's quality.

This paper argues that the link-based Pagerank algorithm is easily biased to produce "unjustified" ranks for low-quality pages. It suggests that the rank of a page should be partly determined by the relevance of the page's content to the theme of the pages that are pointing to it. It is claimed that this approach will produce results that more accurately reflect the importance of pages to particular topic.

Few theoretical approaches for implementing ranking algorithms rely on analysis of content. There is a common assumption to all of them: pages should not be ranked by their content only, but also by the content of the pages that are linked to them.

This paper reviews two algorithms that incorporate content measures in Pagerank computation. These approaches, described in the "Topic-Sensitive PagerRank" [Hav02] and in "The Intelligent Surfer" [RD02] papers, reported the achievement of higher quality search results than the standard link-based algorithm. The downside of both these approaches is their poor computation time performances, which makes both of them inapplicable for practical use.

This project introduces a content-sensitive Pagerank algorithm. The algorithm calculates the rank of pages not only on the basis of the global "popularity" of their inbound-linked pages, but also on the basis of the content match of the inlinked pages. The intuitive justification for the algorithm is that links from pages with similar content should have a larger influence on a page's rank than links from unrelated pages.

The strengthening of "related" links (links connecting thematically related pages) and the weakening of "unrelated" links (links connecting unrelated pages) will cause the grouping of pages into subject-related areas. This project aims to

show that manipulating the propagation of Pageranks in this way can improve the equation performances of the Pagerank algorithm, as the concentration of Pageranks in the center of subject-related groups partitions the computation.

By causing the grouping of thematically related pages, high ranked pages will be positioned in the centers of their respective content related groups.

Figures 1.1 and 1.2 illustrate computation results of link-based and content-sensitive Pagerank algorithms performed on the same Web-graph. Note that pages with inbound links from pages with similar (or related) content (illustrated by the colours) gain higher rank on the content-sensitive approach as compared with the link-based approach.

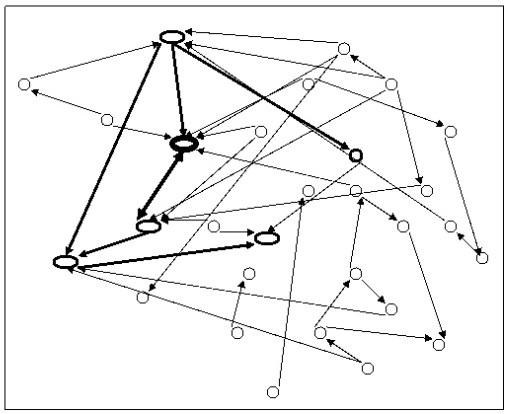

Figure 1.1: Link-based ranking (thickness of nodes represents their ranks, arrows represent links)

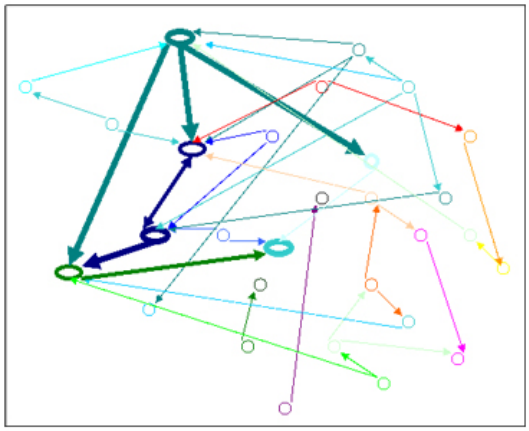

Figure 1.2: Content-sensitive ranking (colours represent content)

The advantages of the distributed Pagernk algorithm proposed, concerning the availability of pages to the ranking agents, open up possibilities for combining content-sensitive measurements with a low cost overhead. This can as well be used to reflect the dynamic behaviour of large parts of the Web, where the content of pages is changes frequently.

1.4 Objectives

The aims of this research can be stated as follows:

- to explore the characteristics of the centralized Pagerank algorithm
- to investigate the aplicability of distributed Pagerank algorithms
- to implement a distributed Pagerank algorithm that can cope with the scale of the Web
- to implement a distributed content-sensitive Pagerank algorithm for improving ranking quality and computation performance
- to suggest ways of reducing communication overhead in the distributed Pagerank computation
- to review techniques for classifying semantic content of documents

- to investigate and develop techniques for reducing rank manipulations (spam)
- to explore and compare the convergence behavior of the two algorithms
- to perform various experiments with both algorithms in a dynamic hypertext environment

Chapter 2

Pagerank Algorithms

2.1 Pagerank: The Random Surfer

Taking advantage of the link structure of the Web, the Pagerank algorithm determines the relative score of web pages by computing the dominant eigenvector of a matrix iteratively. The basic intuition behind Pagerank is that a page should be ranked highly if "the sum of the ranks of its backlinks is high" [BP98]. This intuition covers both the cases when a page has many inlinks and when it has high-ranked inlinks.

Another intuitive way to understand Pagerank is in terms of random surfing. Imagine a surfer who randomly jumps from page to page: the probability of such a surfer visiting a particular page depends on the page's inbound links structure.

The heuristic underlying this approach presumes that in hypertext environments pages having many inlinks are more likely to be of high quality than pages having few inlinks, given that the author of a page will normally include some links to other pages that he or she believes have related content.

2.1.1 Definition

The following definition is based on Page and Brin's paper [PB98]. Simple ranking definition:

$$R(u) = c \sum_{v \in Bu} \frac{R(v)}{Nv} \tag{2.1}$$

Let u be a web page, Fu the set of outlinks from u, Bu the set of inlinks to u. Let Nu $= |Fu|$ and let c be a normalization factor. A rank of a page is the sum of all pages' ranks, divided by the number of outlinks of each page which points to it. The equation is recursive; it is computed by starting with an initial

rank value and iterating until it converges.

c must be smaller than 1, as there are cul-de-sac pages whose weight is lost from the total equation. In this way, rank which is lost due to cul-de-sac pages is redistributed uniformly to all pages in the graph.

The above equation (2.1) is not yet complete. Consider a case where two pages point only to each other and some other pages point to one of the two. Solving the equation this way, these two pages will accumulate Pagerank but will not propagate forward. This loop is called rank-sink [PB98]. To overcome this problem, a vector E, which corresponds to a source of rank, is added to the equation:

$$R'(u) = c \sum_{v \in B_u} \frac{R'(v)}{Nv} + (1-c)E(u) \tag{2.2}$$

E is reported by Brin and Page to either be uniform over all web pages with a value α, or personalized to a certain class of user. The uniform approach, teleporting probability to all pages in the graph, completes the intuition of the random surfer who periodically jumps to a random page. In this sense, when getting into a rank-sink loop, the "random surfer" will eventually jump to a random page in the web-graph.

2.1.2 Centralized Implementation

Google's Pagerank calculation is done centrally. The process can best be explained by dividing it into three main pre-search phases: crawling, indexing, and ranking. Google uses crawlers in parallel, each of which periodically downloads pages into a local database. Google maintains a link structure representation by indexing a large part of the Web. The next step is computing the Pagerank to all indexed links. For a page's calculation, its existing Pagerank (if it has any) is abandoned and a new Pagerank is calculated based on the new link-structure representation of the Web-graph. Pagerank for all indexed pages is then iteratively calculated until all rank values have converged.

The computation of Pagerank, as introduced in Page and Brin papers [BP98] and [PB98], is explained as follows:
Let S be almost any vector over Web pages, and let A be a square matrix representing the link connection of the web pages.

```
R0  <- S
loop{
     Ri+1 <- ARi
     d <- || Ri || - || Ri+1 ||1
```

```
        Ri+1 <- Ri+1 + dE
        delta <- || Ri+1 - Ri ||1
    }while (delta < epsilon)
```

The epsilon value used by Page and Brin is not reported. Nevertheless, epsilon value of 10^{-4}, which is used in various papers (such as [SSW03]and [KSB03]), is reported to have achieved accurate results. Experiments reported in this paper verify this.

As has been reported by Page and Brin [PB98], computing Pagerank for large scale Web-graph (322 million links) converges to a reasonable tolerance in roughly 52 iterations. The fact that the equation over half of the graph converged within 45 iterations, shows that the Pagerank equation will scale very well even for extremely large web-graphs, as the "scaling factor is roughly linear in log n" [PB98].

Pagerank calculation is very expensive: it takes over a week to run. It has to run over a huge sparse matrix, currently of dimension greater than 4 billion [Goo], and is done several times for every class of user. The average time to complete a cycle of crawling, indexing and ranking is not reported by Google. However, as previously mentioned in the Introduction, a test was performed to examine Google's crawling cycles. The test, which is reported in detail in the Experiments section, shows that Google's crawling cycles vary for different pages: "important" pages such as http://www.CNN.com are crawled frequently, whereas other pages are crawled only once every few months.

As the size of the Web grows, the centralized Pagerank calculation becomes the major weakness of the traditional search engines. The scalability and availability weaknesses of the current implementations drive researchers to look for new alternatives.

2.2 Distributed Pagerank Algorithms

The need for search engines that are scalable with respect to the size of the Web, together with the absence of any effective ranking system for the growing P2P networks, has driven research into distributed algorithms for calculating Pagerank. Recently, some highly effective distributed Pagerank algorithms have been introduced, reporting high quality ranking results that are similar to those of the centralized algorithm.

Computing Pagerank in a distributed fashion is not trivial, and cannot be derived from the centralized algorithm in a straightforward fashion. Google's Pagerank is a synchronous iterative algorithm: each iteration step is dependent of the previous one. Implementing synchronous algorithms in a distributed system can easily lead to failures, as it is extremely difficult to achieve synchronous communication across large-scale networks on account of the communication congestions, delays and failures that characterize any open network.

Another difficulty with implementing a synchronous Pagerank algorithm in a distributed fashion is the need to cope with changes in the link-structure of pages. One of the motivations for implementing a distributed Pagerank algorithm is to achieve availability of ranking, i.e. to supply an up-to-date ranking of pages. It can be argued that synchronous algorithms cannot attain this goal. The link structure of pages, as well as their content, is constantly changing. Algorithms based on synchronization of iterations may well achieve high quality of ranking in a static environment, but they cannot handle changes in link structure that occur during execution (such changes include adding/deleting pages, editing pages, or changing their link structure). In synchronous computations, these changes will affect the general outlook for iteration completion. Propagation of Pagerank in a dynamic environment becomes complicated and expensive, and therefore impractical in large scale web-graphs.

A possible way of synchronizing iterations in a distributed system is by using a global clock timing. In such an implementation, every node in the system will propagate Pagerank update messages periodically, giving a fixed time to accumulate rank updates. This approach may be implemented successfully in controlled or small distributed systems. It is much harder to implement in large scale and global distributed systems, as it cannot cope with the congestions and delays that are natural characteristics of a network such as the Web.

The aim of coping with large scale Web graphs and reflecting the dynamic behavior of the Web leads to a different type approach: the chaotic asynchronous Pagerank algorithm.

2.2.1 Distributed Chaotic Asynchronous Pagerank Algorithm

The emerging popularity of P2P networks, together with the absence of effective ranking systems for such networks, prompted research on distributed asynchronous Pagerank for P2P networks.

2.2.2 Existing Pagerank Algorithms

In this section, approaches for implementing asynchronous Pagerank in distributed networks, such as P2P and the Web, will be discussed.

P2P networks introduce new challenges for ranking documents. The dynamic character of P2P networks arises from the fact that peers, normally personal computers, are not constantly connected to the network. Furthermore, peers are not permanently part of the network (someone may disconnect a PC or add another PC to the network at any time). Solutions for ranking documents by a different method have had to be found.

Sankaralingam, Sethumadhavan and Browne [KSB03] introduced a fully distributed implementation of an asynchronous Pagerank algorithm for P2P that attempts to overcome the challenges mentioned above. They developed a method for generating Pagerank values in cases where the matrix representing the link structure of the web graph is never explicitly generated. The method was designed in the context of keyword search for P2P networks but could be implemented in a similar way for the Web. The implementation is based on the chaotic (asynchronous) iterative solution of linear systems. They reported promising results with regard to convergence performance (1% accuracy in 10 iterations) for large systems. [KSB03]
The major phases of their implementation is described below:

1. Every peer initializes all of its documents with initial Pagerank value.
2. Each peer, for every document in its domain, sends Pagerank update messages to all documents which are pointed out (outlinked by the peer's documents).
3. Upon receiving an update message for a document, the receiving peer updates the document's Pagerank.
4. Update of Pagerank of a document in the system results in generation of further Pagerank update messages for all outlinks (propagation of Pagerank).
5. The difference between successive Pagerank values of a particular document is used as a measure to determine convergence of that document. Upon convergence, no more messages are sent for that document.

Sankaralingam et al performed a range of experiments measuring convergence performance over some large graphs of peers. They compared their distributed system Pagerank results with a centralized algorithm similar to the one introduced by Page and Brin, and reported on very low relative error threshold compared with a centralized algorithm.

Experiments performed on 500 peers, using 10K to 5M number of documents, and epsilon value of 10^{-4}, showed impressive yet controversial results (figure 2.1).

graph Size	# of passes
10K	74
100K	88
500K	118
5M	120

Figure 2.1: Distributed P2P asynchronous algorithm, convergence performance results

These results correlate with the convergence results reported by Page and Brin [PB98], in the sense that the scaling factor is roughly linear in log n.

Note the use of "passes" as equivalent to iterations in a centralized algorithm. Sankaralingam et al did not give any explanation or justification for using an iterative measure in an asynchronous algorithm. It could be argued that using a passes (or iterations) measure in a chaotic asynchronous algorithm is problematic. Using iterations in a distributed algorithm must rely on some synchronization to determine the phases of iterations. Since Sankaralingam et al did not discuss the methodology of iterations, it is difficult to discuss their reported results.

Sankaralingam et al rely on a chaotic iterations linear equation solver which is based on a study by D. Chazan and W. Miranker [CM69]. Chazan and Miranker studied chaotic relaxation, also called asynchronous relaxation, for the solution of linear systems. In chaotic relaxation, the order in which components, such as nodes in a distributed system, are updated is arbitrary, and the past values of components that are used in the updates are also selected arbitrarily. This is intended to model parallel computations in which different processors work independently and have access to data values in local memory. The chaotic model of Chazan and Miranker is powerful because of the great generality allowed. As a result, a simple necessary and sufficient condition is needed for the system to converge with chaotic updates [Str97].

As Sankaralingam et al rely on the model of Chazan and Miranker, a likely assumption is that they used some synchronous method to determine an it-

eration unit, and that the asynchronous factor in the system was the non-deterministic order in which update messages ("passes") are sent between the nodes.

If Sankaralingam et al did not use a time synchronization methodology to determine iteration units, the alternative would seem to be that they determined them by maintaining information in the system on the number of inbound links. In terms of implementation, however, maintaining inlinks information for every document in the system is impractical in large scale systems. Moreover, relying on inbound-links information in distributed networks such as P2P and the Web contradicts the true essence of such dynamic systems: informing all the pages in the system that are pointed to by a page of the addition or deletion of a page, or of the changing of the link structure of a page, is impossible in large scale systems.

Whether Sankaralingam et al maintained inbound-links information in their system remains unclear from their paper. Later on in this chapter, the parameters of this problem and suggested solutions to it will be discussed.

In the paper "Distributed Page Ranking in Structured P2P Networks" [SSW03], ShuMing Shi, Jin Yu, GuangWen Yang and DingXing Wang introduced two distributed Pagerank algorithms (DPR1 and DPR2) that rely on the fact that for most pages in the Web (as well as documents in P2P networks) inlinks are normally pointed to by physically close sources (usually pages located on the same web-server). This assumption led to the design of distributed iterative algorithms adopting aspects of the centralized Pagerank algorithm introduced by Page and Brin.

However, it can be argued that relying on the above assumption is inadvisable. There are many important pages, such as www.yahoo.com, www.google.com, and www.bbc.co.uk, that gain their high ranking from having thousands of other pages pointing to them, many of which are not physically close to the page in question. As mentioned earlier, an iterative based distributed algorithm cannot be implemented successfully in large scale Web environments, as the scalability and availability of such environments cannot be reflected by a ranking algorithm that requires any aspect of synchronization during the ranking equation.

2.2.3 Controversial Results of Equations Performances

Sankaralingam , Sethumadhavan and Browne introduced high quality Pagerank results compared with a centralized algorithm. They examined the quality of the Pagerank values generated by the distributed computation by measuring the relative error in Pagerank, using the formula *abs(Pd - Pc)/Pc*, where *Pd* is the a Pagerank value of the distributed computation for a certain page and *Pc* is its Pagerank value which was computed in the centralized algorithm.

They reported that in practice, Pagerank Pd "converged to within 0.1% of Pc in as few as 30 passes" [KSB03]. Furthermore, they observed that for all the graphs, "more than 99% of the nodes converged to within 1% of Pc in less than 10 passes" [KSB03]. Sankaralingam et al give a very detailed table, testing different epsilon values and error thresholds, which shows excellent results in Pagerank quality for a big data-set of 5000k nodes.

These results may not be completely reliable. A closer look at their report reveal that Sankaralingam et al determine a convergence of a page by comparing the relative difference of successive ranks of that page, instead of the absolute difference:

```
relerr = abs(oldrank - newrank)/newrank;

if(relerr > epsilon){
    send pagerank update messages to all outlinks
}
```

As shown, when the relative error threshold value of a page is higher then the epsilon value (a non-convergence stage), update messages are triggered.

In Page and Brin's algorithm [PB98], convergence of a page is measured by comparing the absolute difference of successive ranks to the epsilon value:

```
loop{
    ...
    delta <- || Ri+1 - Ri ||1
}while (delta < epsilon)
```

Computing the relative difference of successive ranks for a given page will obviously result in a lower value compared with the absolute difference of the same page. Hence, when using the relative difference in a Pagerank, the computation of the collection of pages will reach convergence much faster. As a result, the overall ranking values of such algorithms will be very inaccurate. Highly ranked pages, which usually serve as the major source of rank for large collections of pages, will stop propagating Pagerank values prematurely. This behavior will lead to an unbalanced distribution of Pagerank, and, as a result, an inaccurate ranking graph: pages which are linked by other "important" pages will not gain their relative share of rank, as the contribution of high-ranked pages will be limited by their disproportionally fast convergence.

2.3 Incorporate Content Analysis in Pagerank Computation

The motivation for developing distributed Pagerank algorithms is mainly derived from the need to achieve scalability and the lack of a ranking system for P2P networks. It may well motivate research into the reevaluation of a rather controversial subject: content analysis of hypertext pages as a ranking criterion.

Within the last couple of years, content homogeneity of web-sites has been discussed as a possible criterion for ranking pages. There are few theoretical approaches to implementing algorithms for search engines that rely on analysis of content. There is one theme common to all of them: pages should not be ranked by their content only, but also by the content of other linked and related pages. The potential implementation of content-sensitive ranking that is combined with a centralized ranking system such as Google, is discussed controversially in search engine optimization forums [SEO].

It has been argued that using the link structure of the Web as a single measure for determining the relative "importance" of pages is inaccurate and insufficient for achieving good search results. It is claimed that the Pagerank algorithm is an insufficient ranking scheme, which may only evaluate the general popularity of pages rather than their quality.

The relevance of a page to a query, it is suggested, must not be determined mainly by the page's general popularity. Other factors, such as the position of pages in an environment that is correlated to certain key-words, should have a high weighting in determining the ranking of pages.

As the main weaknesses of the centralized Pagerank algorithm are scalability and availability, it is argued that adding content analysis to the computation in order to improve rank quality will increase these problems. The next sections discuss this hypothesis.

2.3.1 The Intelligent Surfer

One of the first attempts to improve Pagerank using content analysis was made by Matt Richardson and Pedro Domingos. In their paper "The intelligent Surfer: Probabilistic Combination of Link and Content Information in Pagerank" [RD02] Richardson and Domingos proposed a formalized behavior pattern of a "more intelligent surfer, one that is guided by a probabilistic model of the relevance of a page to a query" [RD02]. They suggested a type of surfer that only follows links that are related to the terms of its query, as opposed to Page and Brin's Pagerank algorithm which is described in terms of random surfing.

The link-based Pagerank algorithm rates a page highly if it is at the center of a large sub-web (i.e., if many pages point to it, many other pages point to those, and so on.). Intuitively, however, "the best pages should be those that are at the center of a large sub-web relevant to the query" [RD02]. Domingos and Richardson propose a surfer who will normally jump to pages that contain terms from his or her original query, instead of randomly surfing. This makes sense because, for example, in a query containing the word "jaguar", pages containing this term that have many links from other pages that also contain this word (and so on) must be more relevant to the query than pages that contain this term but do not have other pages pointing them that contain the term.

Richardson and Domingos proposed a search algorithm that formalizes this intuition, which, as in the Pagerank algorithm, does most of the computation in crawl-time. They proposed QD-Pagerank (Query-Directed Pagerank), a separate Pagerank computation that is to be done for every individual term that occurs in the Web. In other words, for every term in the Web, a QD-Pagerank vector will be calculated. The search mechanism works by executing a query on the combination of terms-vector which correlate to the query terms.

In such a computation, the Pagerank score can be viewed as the rate at which a surfer would preferentially visit pages containing the query terms.
The resulting distribution over pages (for a query q) is given by:

$$Pq(j) = (1\text{-}c)P'q(j) + c \sum_{i \in B_j} Pq(i)Pq(i \rightarrow j) \tag{2.3}$$

Let $P'q(j)$ specify where will the surfer jump when not following links. Let $Pq(\mathrm{i} \rightarrow \mathrm{j})$ be the probability that a surfer transit to page j given that he is in page i.
$Pq(j)$ corresponds to query-dependant-Pagerank (QD-Pagerank(j) $\equiv Pq(j)$).
Similar to Pagerank, QD-Pagerank is calculated by an iterative evaluation of equation 2.3.

Let $Rq(j)$ be a function that measures the relevance of page j to a query q (this function can be based on TFIDF measure, for example), then (as shown in 2.4) the arbitrary distribution of Pagerank from page j, is guided by the relevance of links in the entire Web (a guided teleportation):

$$P'q(j) = \frac{Rq(j)}{\sum_{k \in W} Rq(k)} \tag{2.4}$$

Let W be all pages in the graph.
Equation of $Pq(j \rightarrow j)$ can be described in a similar way(2.5):

$$Pq(i \rightarrow j) = \frac{Rq(j)}{\sum_{k \in Fi} Rq(k)} \tag{2.5}$$

Where Fi represents all forward links from i.

Explaining this in words, when a surfer chooses a link from a given page, he will tend to follow those which are relevant to the term he or she is looking for (relevant to a query). Similarly, Pagerank distribution from page i in the QD-Pagerank algorithm will be directed to pages which are relevant to the query. When there are no "relevant" links from that page (or no out-links from that page), the content-directed surfer will "jump" to other pages in the graph that are relevant to the term he or she is interested in.

As mentioned above, Richardson and Domingos suggest pre-computing a rank-vector for every term that appears in the Web, in other words, computing a different QD-Pagerank for every term. Then, when a query is made, a search will be performed on the corresponding rank-vector, or, in a multi-terms query, on the relevant combination of rank-vectors.

Richardson and Domingos are aware of the scalability challenges. They reported that computation and storage requirements for "**hundreds of thousands of words is only approximately 100 times that of a single query independent Pageant**" [RD02].

An important aspect of Richardson and Domingos' paper is the successful results they achieved in "human" evaluation of search results.
They compared results of queries in executed on a link-based Pagerank vector to a QD-PageRank vector by letting volunteers rank the level of relevance of their queries in both systems.

The results reported for the OD-Pagerank algorithm are:

- Average of 20% better results in *edu* Web-graph
- On average of 34% better results in *WebBase* graph (Stanford University experimental large scale Web-graph)

Considering the centralized Pagerank algorithm's main weaknesses, which are availability and scalability, implementation of QD-Pagerank, which increases the problem one-hundred-fold (not only in terms of time but also in disk-space and memory requirements) becomes impractical. Apart from that, computing QD-Pagerank may cause some other problems.

Firstly, the number of terms that appear in the Web is much greater than 100,000. One of the experiments described in the Experiments chapter of this paper, which required generating a term-vector from 35,000 pages of mainly pages with English textual content, resulted in allocation of more then 225,000

different terms. Taking into account the scale and the variety of languages represented in the Web, the number of terms to be calculated differently in the QD-Pagerank algorithm is much higher than Richardson and Domingos estimation.

Secondly, to make it into the Pagerank calculations for a specific term, that term has not only to appear on a certain page, but should also appear on pages that link to it. In this sense, the search results would often be based on small subset of the Web, and may omit relevant web pages. In addition, QD-Pagerank is in that sense more vulnerable to creating spam, as manipulations of small subsets of the Web are easier to perform.

2.3.2 Topic Sensitive Pagerank

In his paper "Topic-Sensitive Pagerank" [Hav02], Taher Haveliwala introduces an approach to combining link structure and content analysis that seems more practical for actual implementation than that of Richardson and Domingos. Indeed, there has been some speculation in the Search-Engine-Optimization community as to whether the Topic Sensitive Pagerank algorithm may be implemented in Google's search engine [SEO].

Like Richardson and Domingos [RD02], Haveliwala proposed a computation of different Pageranks guided by different content. However, Haveliwala's algorithm does not perform hundreds of thousands of Pageranks computations for different terms, just a few Pagerank calculations for different topics.

"We compute off-line a set of PageRank vectors, each biased with a different topic, to create for each page a set of importance scores with respect to particular topics" [Hav02](p. 2). The intuition behind this approach is that a page that is considered important in one subject may not be considered important in another. Havaliwala implies that there should be a different weight for every topic when a search is performed.

Instead of ranking pages according to a universal popularity measure, as the link-based Pagerank does, Havaliwala differentiates ranks on the basis of different topics.

Havaliwala first defines authority pages for each topic. These are assigned a relatively high Pagerank value E (as a bonus). He then performs a different Pagerank computation for each topic, including the entire Web-graph in each computation, to create Topic-Sensitive Pagerank vectors which will be used later on in search algorithms. By doing this, Haveliwala is adding a determinative "power" to the URLs that appears in the ODP: they become a major source of rank which influence the ranking of the entire Web-graph.

Haveliwala suggested using the Open Directory Project [ODP] database, which contains currently more then **4 millions URLs**, divided to **16 topics**. The Open Directory Project is the largest, most comprehensive human-edited directory of the Web. It is constructed, evaluated and maintained by a vast, global community of **64,443 volunteer editors**. Haveliwala's justification for using the ODP: "One could envision using other sources for creating topic-sensitive PageRank vectors; however, the ODP data is freely available, and as it is compiled by thousands of volunteer editors, is less susceptible to influence by any one party" [Hav02] (p. 2).

To compare the query-sensitive approach to ordinary Pagerank, Havaliwala conducted a user study test. He randomly selected 10 queries from a test set for the study, and found 5 volunteers to evaluate the search results. [Hav02].

Havaliwala acknowledges the disadvantages of choosing ODP for identifying topics, which for one thing results in high dependence on ODP editors and a rather rough subdivision into topics [Hav02].

The results reported by Havaliwala are very positive in the Topic-Sensitive algorithm. For **80%** of the queries, the volunteers chose the Topic-Sensitive results over the No-Biased results.

A crucial point in Havaliwala's architecture is the need for identification of the correct topics when performing search queries. Implementing such functionality is not a trivial task, and although Havaliwala suggested some techniques that might accomplish it, they may be considered controversial as they may reduce performance in terms of search time. As searching algorithms are not the main focus of this paper, this fascinating aspect of Havaliwala's work cannot be discussed here.

Although Havaliwala's Topic-Sensitive approach is much more practical than that of Richardson and Domingos it cannot cope successfully with the growing scalability problem of the centralized Pagerank algorithm, as it has to compute Pagerank 16 times for the entire Web-graph. Doing such computation in parallel is possible, but is unlikely to be economic.

2.4 Distributed Content-Sensitive Pagerank Algorithm

That it is possible to weight links based on representative topic has been shown in the "Topic-Sensitive Pagerank" paper. This project suggests using content analysis of pages in the Pagerank algorithm. The motivation for this approach is to eliminate unjustified rank from the equation. Links within a page that are linked to thematically unrelated pages should have only minor influence on that

pages rank. It is desirable to capture more accurately the notion of importance with respect to a pages' theme.

Encouraged by the improvement in search results reported both by Haveliwala and Richardson and Domingos, together with the performance challenges raised, an attempt to examine the effectiveness of introducing a content-sensitive measure in a distributed Pagerank algorithm is studied in this project.

2.4.1 Weighting Links Based on Content Analysis

The thought behind weighting links based on content analysis has been discussed in Search-Engines-Optimization forums [SEO] for the last couple of yeas. The idea of weighting links based on the content relation of the interconnected pages, is mainly discussed as a technique of avoiding the corruption of the link-based Pagerank. Diminishing the influence of links between thematically unrelated pages, which have been set for the purpose of boosting Pagerank of a pages, is theoretically possible but it raises two important questions:

1. Can the overhead of content analysis be tolerated in practice?
2. Is it justified to reduce the influence of links to pages which contain general content or to pages with little textual content (such as google.com)?

To answer these questions, this project suggests combining content analysis with the link-based Pagerank whilst giving higher weight to the link-based aspects of the equation. This way, "popular" pages would preserve a relatively higher rank.

Performance shortcomings caused by the overhead of analyzing content of pages can be eliminate by using the natural partition of the Web to Web-servers and the immediate access of Web-servers to local pages.

2.4.2 Vector Space Model and TFIDF

Processing text electronically was first introduced in the late 60s by Gerard Salton. Salton developed a formula for evaluating the importance of terms within a document. Salton's vector space model [SS95] of information retrieval is based on generating term-vectors for documents - sequences of term-weight pairs appearing in documents.

The vector space model of text processing is the most widely used information retrieval model. In this model, every information item - including the stored texts and any natural language information request- is stored in a vector of terms. These terms are usually extracted from a collection of text sources.

Words are reduced to their morphological roots, or words stems, using a well-defined set of rules [SS95].

Since different terms have different importance in a document, an indicator - the *term weight* - is associated with every term. Higher weight is assigned to more important terms. Typically, a term that occurs frequently in a text is more important in the text than an infrequent term.

Term importance is not independent of the examined document. The more documents a term occurs in, the less important it may be. So an *inverse document frequency* (or idf) factor is incorporated into term weights [SS95].

The TFIDF measure can be calculated using two different approaches. The algorithm used in this project is described in equation 2.6.

$$W(i,j) = \frac{tf(i,j)}{idf(i)} \tag{2.6}$$

For a term i in document j, let $tf(i,j)$ be the number of occurrences of i in j, and let $idf(i)$ be the number of occurrences of i in entire collection.

Long term verbose documents usually use the same terms repeatedly. As a result, the term frequency factors may be large for long documents. Long documents also have numerous different terms. This increases the number of word matches when comparing document content. To cope with these effects, a normalization of term weight is used. Such normalization imposes a penalty on the term weights for longer documents [SS95].

Using the TFIDF measure for comparing content of pages can be done by computing the inner product overlap of terms. This technique is used in search engines for finding the most relevant document to a query (which is in the form of term-vector).

Using TFIDF for comparing content of pages is argued to be effective for most cases. However, there are some cases where content comparison is not sufficient to determine their relation, for example: pages containing no textual information, pages containing similar themes written in different languages, and pages introducing similar theme using different vocabulary. Stemming techniques for normalizing term vectors have been proven to achieve high quality term-vectors which can be used for comparing content of pages [MKM00].

The vector-space-model is widely used as a basis for successful algorithms for document ranking, document filtering and document clustering.

2.4.3 The Term Vector Database

A successful implementation for comparing Web-pages, based on Salton's vector space model, has been introduced by Krishna Bharat, Farzin Maghoul, and Raymie Stata, in their paper "The term vector database" [KBS00]. They built a database that provides term vector information for large numbers of pages (hundreds of millions) [KBS00]. The database is used as a platform for several applications, such as applications for optimizing connectivity-based topic distillation, and Web page classifiers used for annotating results returned by a web search engine.

An interesting feature of the Term vector database, which could be useful for content comparison across large collection of pages, is the lexicon used for filtering out insignificant terms from term vectors. The lexicon was built by taking the "middle third" terms from the AltaVista index minus a few dozen terms from a stop list. That is, they eliminated the most frequent third and the least frequent third. Taking out the most frequent terms is done because they provide little discrimination across vectors [KBS00]. The least frequent terms are noisy and do not provide a good basis for measuring semantic similarity. For example, one such term is *hte*, a misspelling of *the*. This term appears in a handful of pages that are completely unrelated semantically. However, because this term is infrequent, its appearance in term vectors makes those vectors appear to be quite closely related.

2.4.4 Combining Link-Structure and Content-Analysis

This project suggests combining link-based Pagerank algorithm with content analysis of pages. It is suggested that basing a ranking algorithm on both the link structure and the content of pages will provide more balanced ranking results. For example, measuring Web pages solely according to content analysis would not be effective for pages that contain little textual content.

The idea of weighting links based on content analysis has been discussed for some years in the context of avoiding the corruption of Pagerank algorithm. By weighting links this way, it is theoretically possible to diminish the influence of links between thematically unrelated pages, which have been set for the purpose of boosting Pagerank of one page [EFT].

This project suggests using comparison of pages as a factor in the Pagerank equation. A page's rank would be strongly influenced by other pages pointing to them which are content related.

The grouping of thematically related pages is the motivation of ranking pages this way. By doing this, highly ranked pages would be the pages which are in the center of their content related group.

This paper argues that an effective content-sensitive ranking system could be built on top of a distributed Pagerank algorithm. The advantages of the distributed Pagerank algorithm proposed, concerning the availability of pages to the ranking agents, opens possibilities for combining content-sensitive measurements in a low cost overhead.

The advantage of having immediate access for local pages in a distributed system based on Web-servers that it may be used to reflect the dynamic behavior of large parts of the Web, when the content of pages is frequently changed.

Partitioning the Web caused by rank grouping of closely related pages could achieve higher convergence performance of computing Pagerank compared with link-based computation. The reduction of "unjustified" ranks, together with the rank grouping behavior of a content-sensitive algorithm, could help in overcoming the poor computation performances reported by previous attempts of introducing content analysis into the Pagerank equation, as reported by Haveliwala's and Richardson-Domingos' papers ([Hav02] and [RD02]).

Chapter 3

Software Architecture

Designing a distributed system for computing Pagerank is a challenging task that faces two major difficulties: managing large quantities of data and handling communication overhead. To meet this challenge, this project presents a non-iterative asynchronous Pagerank algorithm that relies on the structure of web-servers. The main component of the system, the Pagerank Agent, is designed as a daemon to be run attached to a web-server, to represent the web-server's pages in a collective calculation of Pagerank.

This chapter introduces a distributed system for calculating link-based and content-sensitive Pagerank using up-to-date technologies. High flexibility is achieved using an Object-Oriented methodology and a configurable-based approach in the overall design. Having a flexible and extendable system is important for testing different ranking algorithms.

The resulting implementation allows experimentation with a variety of content-sensitive Pagerank algorithms, using a simple configuration file for setting the overall behavior of the system. A snapshot mechanism is used for plotting statistical information of the Pagerank computation in run-time.

Java was the programming language of choice, for reasons that will become clear during the course of this chapter.

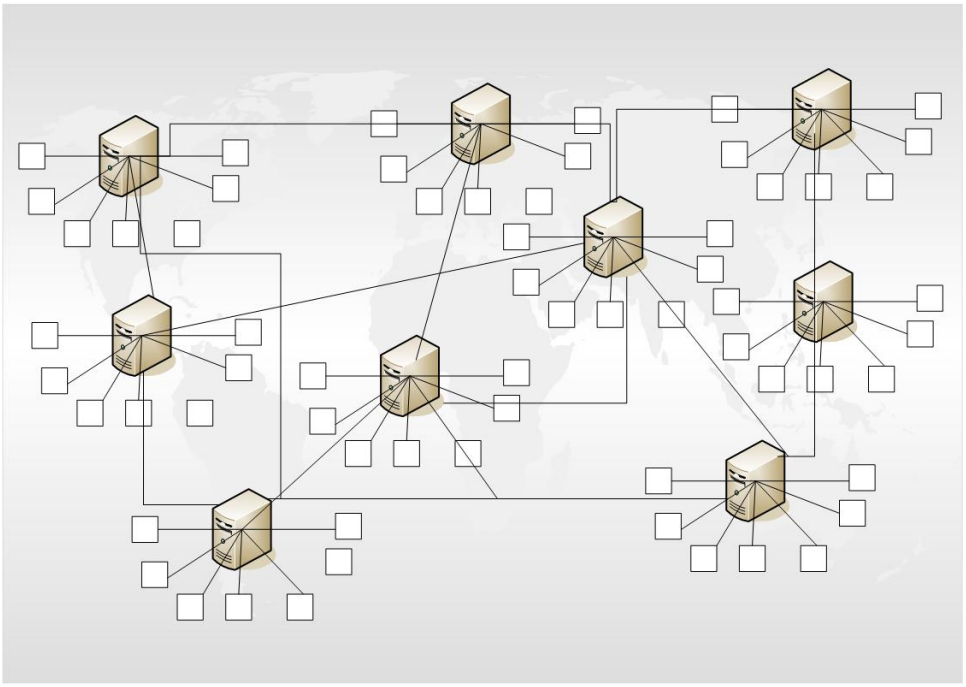

Figure 3.1: Pagerank-Agents running on Web-servers

3.1 Multi-Threaded Pagerank Agent

The main component in the system is the Pagerank Agent. It is designed to run on a web-server as a daemon that handles all aspects of computing Pagerank for the web-pages located on the web-server. As the main aim of the project is to investigate aspects of a content-sensitive Pagerank algorithm, an actual implementation accessing web-servers pages is not presented. Alternative methods of retrieving content and links information on the web-servers' pages are suggested.

The Pagerank Agent is a client-server application designed to be available at all times as a daemon that handles sending and receiving Pageank update-messages for the pages on the web-server it represents. The implementation is based on the Java-RMI technology, using synchronous method invocation for exchanging Pagerank update-messages. Figure 3.2 shows the general structure system.
Pagerank-Agent is combined of 4 main modules:

- Link-structure (in memory) Database
- Multi-threaded RMI Server: receives update-messages
- Multi-threaded Pagerank Updater: generates Pagerank calculations
- Multi-threaded RMI Client: Distributes Pagerank update messages

These modules function as a single unit that handles all ranking aspects, including plotting statistics of the equation progress.

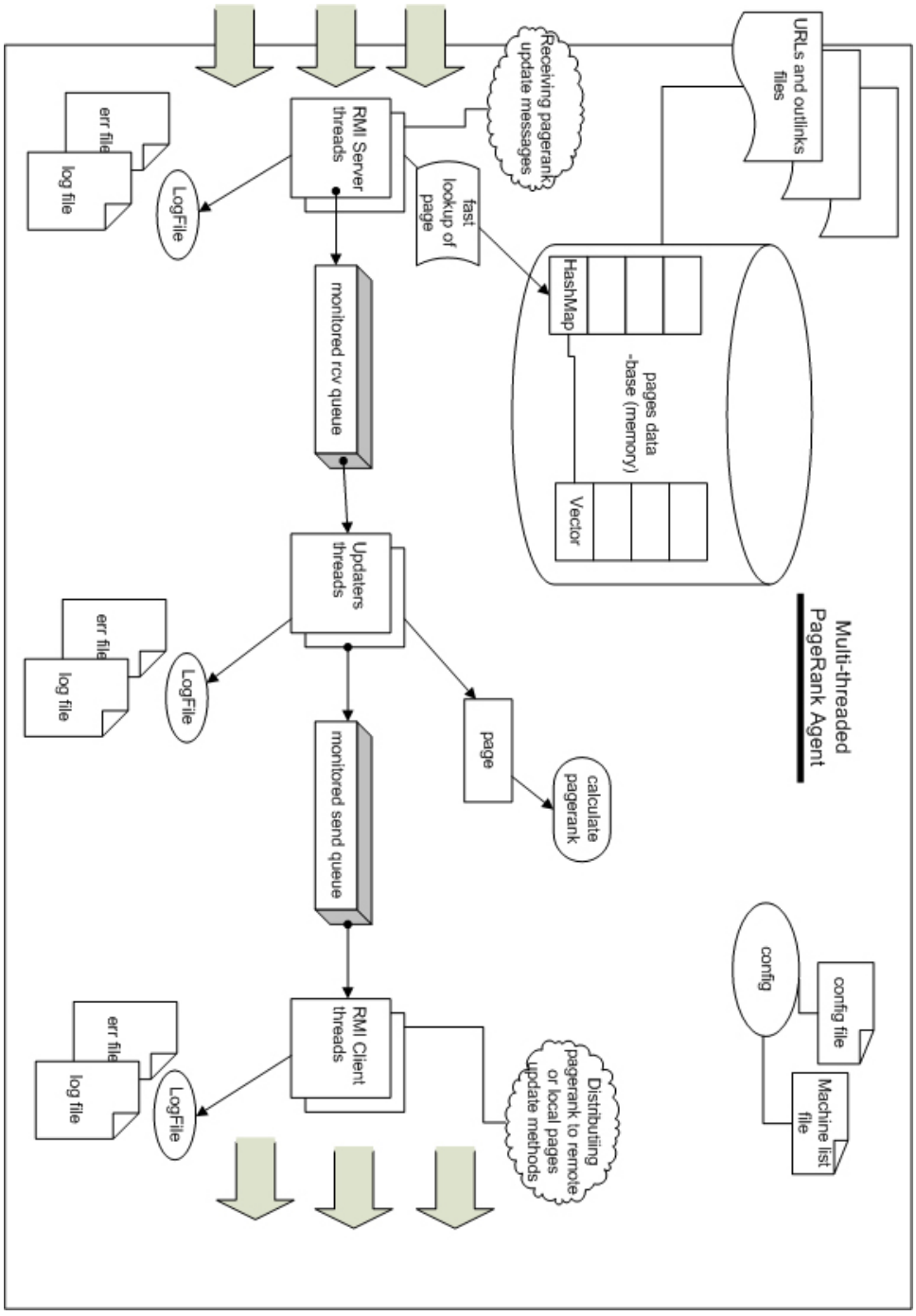

Figure 3.2: Multi-threaded Pagerank Agent

3.1.1 Web-graph Partitioning

Ideally, a Pagerank-Agent would run on a web-server, accessing pages' data directly according to the server's structure. In this project, different ways of retrieving the data are used in order to allow flexible executions of the system in Lab conditions. The Web-graph is artificially split into groups of host-related pages in order to create an environment that correlates to the structure of web-servers.

The configuration of the system is done automatically. As demonstrated in figure 3.3, a file containing a list of RMI-Agent addresses is loaded into all agents participating in the experiment. Every agent in the system reads a web-graph file (containing the link structure of pages) and loads into memory only the URLs that correlate to the agent's ID. In this way, after the initialization process, every agent holds in memory the necessary information on its "share" of the web-graph. The algorithm used to divide the URLs between different machines is explained by the pseudo code below:

```
 for every URL in the web-graph {
    serverID = hash(host name of URL) % (no. of machines in the system)
    if (serverID == AgentID) then load into database
}
```

The additional configurable parameters used in the system are listed below:

- Number of threads to be executed for the Servers, Clients and Updaters
- epsilon value - determines the threshold for triggering Pagerank distribution
- Content-Sensitive measure: TFIDF or Frequency
- Number of terms to be used in the Content-Sensitive comparison
- Content-Sensitive factor - the weighting of links according to content match

3.1.2 Inbound-Links versus Outbound-Links

A major design concern for a distributed Pagerank algorithm is choosing the method for maintaining the link structure of pages. This decision is normally determined by the nature of the algorithm used.

Previous implementations of distributed Pagerank algorithms ([KSB03]) rely on the use of inbound-links in the Pagerank equation. The use of inbound-links in Pagerank equations requires storing information about inbound-links for every page in the system. This paper argues that storing inbound-links for

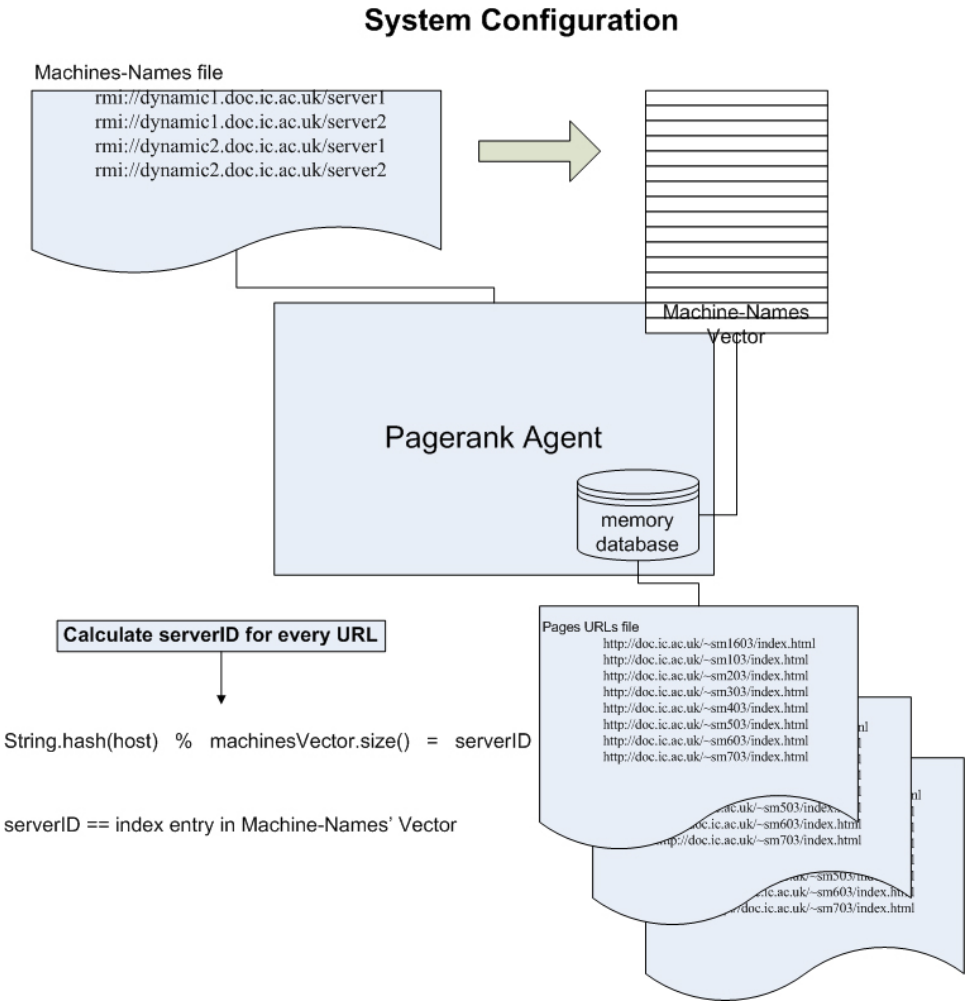

Figure 3.3: Partitioning Web-graph Auto-Configuration

pages is impractical in a distributed Pagerank equation because:

- Popular pages, such as www.yahoo.com, www.google.com, etc. may have an enormous number of inbound-links
- In a large scale distributed system one cannot discover the source of inbound-links of pages

The alternative implementation introduced in this project maintains only the outbound-links of each page. This approach is suitable for the distributed non-iterative Pagerank algorithm since no synchronization of iterations, which would require preserving inbound-links information, is performed.

3.1.3 In-Memory Database

In the final stage of the initialization process, every Agent in the network maintains information on the "local" URLs (pages) and their out-links. Information on outlinks, current and previous Pagerank values and general statistics (such as the number of messages sent and received and convergence time) are maintained for every Page in the system.

All page objects are stored in a HashMap, a standard Java implementation for Hash-Table. Hash-Table, which has a complexity of nearly O(1), is a suitable data-structure for the fast retrieval of information, which is essential for locating Pages in order to update Pagerank values. The HashMap data-structure also allows the viewing of its objects as a list. This functionality is useful for printing page statistics for a collection of pages.

3.1.4 RMI Client-Server Implementation

The synchronous client-server approach is chosen in order to obtain accurate Pagerank updates. An alternative approach using the UDP protocol, which would reduce communication overhead (a drawback of synchronous protocols), was considered. Using the UDP protocol, implemented with acknowledgment messaging for improving reliability, would decrease communication overhead in the system as no connection setup exists. On the other hand, using the UDP protocol would add complexity to the implementation as a mechanism for tracking acknowledgement-messages is not a trivial implementation. Due to time limitations this approach was not implemented.

The RMI object-invocation technology is chosen to attain flexility and simplicity of the architecture. This way one can provide a wider range of possibilities for a variety of experiments.

One of the advantages of distributed asynchronous Pagerank algorithms over synchronous algorithms is the option of calculating Pagerank updates on the sender side. This improves communication performance immensely, as explained in the next sections. The overall flow of the Pagerank-Agent, demonstrated in figure 3.2, is described below:

1. In the initial stage, pages are given the Pagerank value of (1-c). This value is a dumping factor which represents the teleportation distribution of Pagerank.

2. The Initialization process ends by sending all Pages to the send-queue (a monitored queue - designed for thread-synchronization).

3. RMI Clients, awakened by the arrival of pages in the send-queue, then start distributing Pageank-update messages for all pages' outlinks.

4. The binding mechanism, explained in the next sections, triggers calls to either remote or local methods, depending on the outlink's page location. For local pages, a Pagerank-update method is called, whereas for pages located on other agents, the client executes an RMI method-invocation for updating Pagerank values.

5. When receiving a Pagerank-update message, the RMI-Server retrieves the relevant Page by using the target URL (which is included in the message received) as a key.

6. The retrieved Page object, together with the update message, is added to the received-queue.

7. The Updaters threads, awakened by the arriving messages, execute Pagerank calculations by calling a method in the Page object.

8. Updating Pagerank for a Page may trigger forward distribution of Pageranks from that Page. In this case, an update message is to be generated, and as a result a message is added to the sending-queue for Pageank distribution.

9. The convergence stage of the collection of pages is achieved when no update messages are sent.

3.1.5 Updater: Non-Iterative Pagerank Algorithm

Calculating Pagerank in an asynchronous distributed algorithm, as proposed in this paper, is a non-iterative process. The standard centralized Pageank algorithm can be described as follows [BP98]:

PR(A) = (1-d) + d(PR(T1)/C(T1) + ... + PR(Tn)/C(Tn))

Let C be the number of outlinks (i.e. citations) of a page, and assume that page A has pages T1...Tn pointing to it (the parameter d is a damping factor which can be set between 0 and 1).
The equation is iteratively computed until the difference of successive Pageranks is less than epsilon, which is usually set to 10^{-4}.

The asynchronous Pagerank algorithm proposed in this project is not computed iteratively. Instead, asynchronous Pagerank updates for a page are sent by pages that point to it. These Pagerank updates are accumulated and tested on an epsilon threshold, and trigger new distribution of update messages. The following pseudo code explains the equation:

```
upon receiving an update message:
newrank = oldrank + (c * update\_message)

if( (newrank - oldrank) > epsilon ){
    a.) update_message = (newrank / #_outlinks) - (oldank / #_outlinks)
    b.) send update messages to all out-linked pages
}
```

Basing the algorithm on a non-iterative equation may lead to some variations in Pagerank values. That is, running successive executions of the Pagerank equation using the same configuration of the system may end with different Pagerank results. The reasons for this are the characteristics of the asynchronous non-iterative equation and the unreliable behavior of the IP network.

3.1.6 DNS Lookup: Major Performance Problem

Ideally, Pagerank update-messages could be directed from a page to its out-linked pages in a system that runs Pagerank-Agent daemons on web-servers by addressing them to the hosts of the out-linked URLs. For example see figure 3.4.

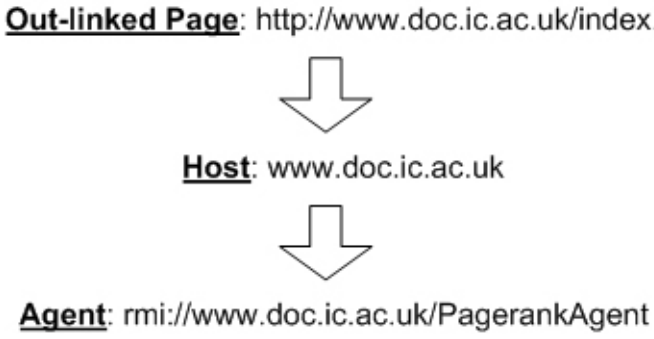

Figure 3.4: Directing a Pagerank-update-message to a Pagerank-Agent

A major problem with this solution is the DNS lookup mechanism. The system is dependant on retrieving IP addresses using the DNS lookup mechanism as the only information gained on outlinks is their URLs. As the average number of outlinks per page is approximately 11 (as reported by Page and Brin in 1998 [BP98] - the figure is probably higher now), whenever a page triggers Pagerank distribution for its out-linked pages, a possible 11 DNS lookups are performed to locate the IP addresses of the Pagerank-Agents. In large scale environments such as the Web this mechanism increases communication overhead immensely.

This paper suggests a caching of IP addresses for the out-linked URLs mechanism. In the first distribution of Pagerank messages generated by a page, all IP addresses of the out-linked pages retrieved by the DNS mechanism are stored in memory attached to the outlinked URLs. In the next distribution of Pagerank update-messages, the system would use the cached IP addresses.

Running the distributed Pagerank system in Lab conditions requires a slightly different mechanism for sending update-messages. Agents are configured with a fixed list of IP addresses that represent the actual machines participating in the experiment.

3.1.7 Cul-De-Sac Pages: Random Distribution

Brin and Page define dangling links as links that point to pages with no outgoing links [PB98]. These links point either to pages which are not downloaded yet or to cul-de-sac pages (pages with no outlinks). In the centralized algorithm (Google), these pages are taken out of the Pagerank calculation as they do not influence the ranking of any other pages. After all Pageranks are calculated, these pages are added back in, without significantly affecting calculations [BP98].

In a distributed system, using a similar solution for cul-de-sac pages is impossible, for the reason that there is no control of the convergence stage of the entire system. Communicating the relaxation of the Pagerank equation between all nodes in the system is undesirable in large scale Web-graphs.

An alternative solution is suggested, namely: as cul-de-sac pages are not removed from the system during Pagerank computation, a random distribution of Pagerank update-messages from cul-de-sac pages can be performed. Sending Pagerank update-messages to random pages will result in an equal propagation of Pagerank across the Web-graph. In this manner no Pageranks are lost in the computation.

The major difficulty of implementing a solution of this kind is the need for "true" randomness when distributing Pageranks across the Web-graph.

This project offers an implementation of random distribution. Assuming an average of 11 out-links per page, when a cul-de-sac page triggers the distribution of Pagerank update-messages, random distribution is performed in the following way: 11 URLs are randomly selected from the collection of outlinks maintained in the local database. Standard Pagerank update-messages are sent for some of the outlinks selected, and Pagerank random-messages are sent for the rest. Upon receiving random-messages, Pagerank-Agents randomly select outlinks from their database and forward the distribution in the same manner.

Increasing the percentage of Pagerank-random-messages sent from cul-de-sac pages will strengthen the randomness of propagating Pagerank values on one hand, and will raise the overall number of messages triggered on the other. Figure 3.5 shows the random distribution of Pageranks.

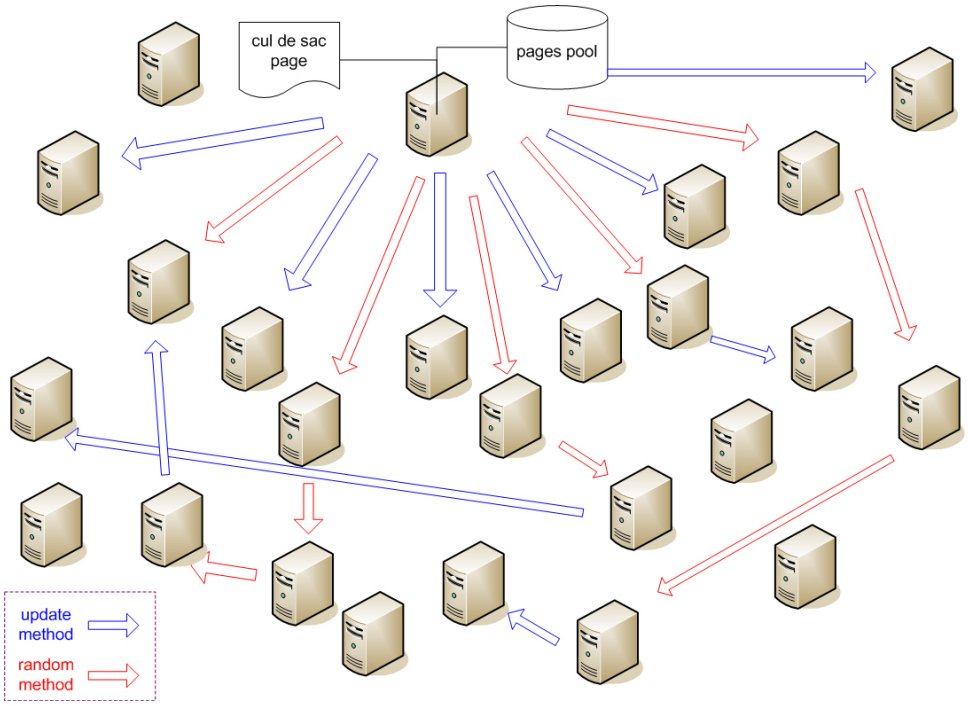

Figure 3.5: Random Distribution of Pagerank using a factor of 50%

The main weakness of such an implementation is the fact that popular URLs, having more inbound-links than others, are more likely to receive random-Pagerank-updates. Because of this, random propagations of Pageank will strengthen pages with high rank value. Such a distribution is similar to the "random surfer" intuition of Page and Brin [PB98]. Popular pages (pages with more links pointing to them) are more likely to be visited by a random surfer.

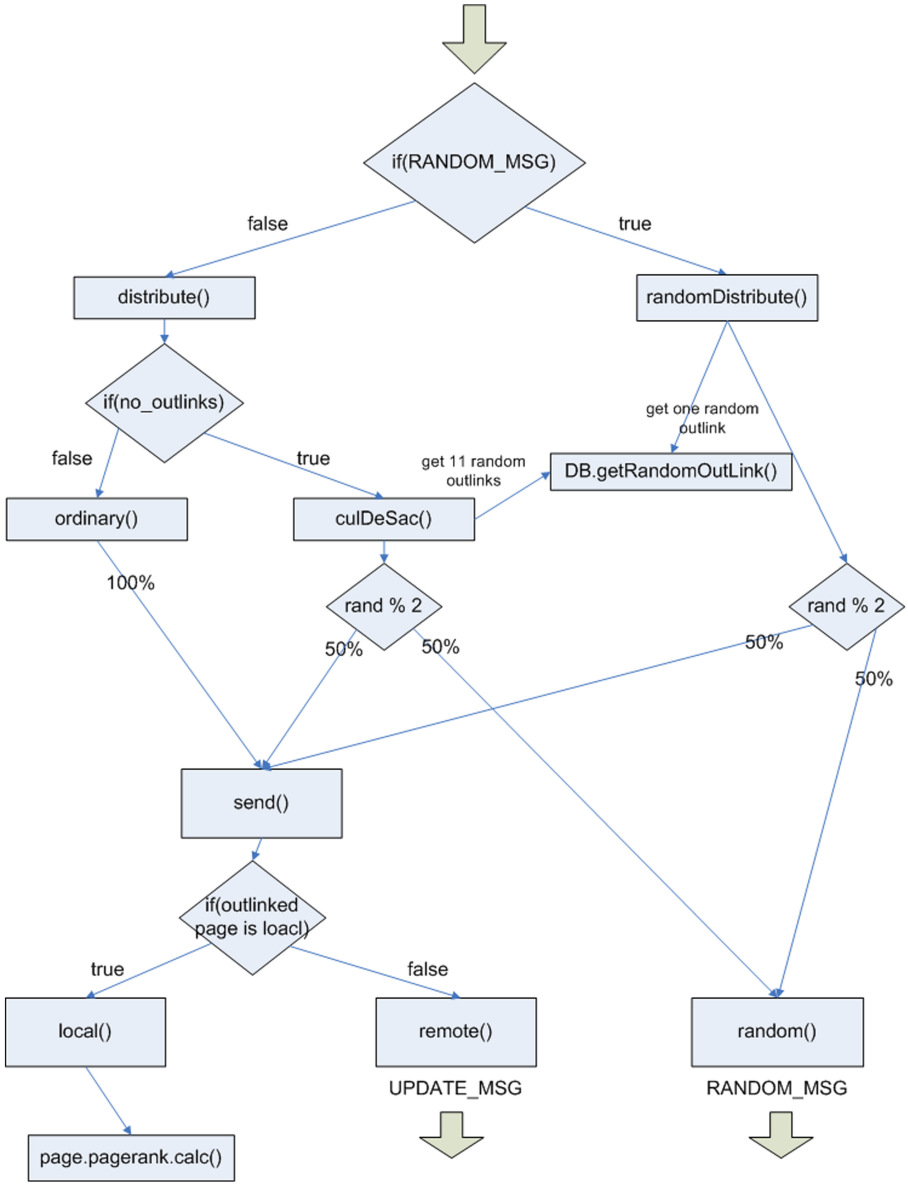

Figure 3.6: Distribution of Pagerank to out-linked pages, RMI-Client Flow Chart

3.2 Content-Sensitive Pagerank Algorithm

The idea behind introducing content analysis into the Pagerank computation is to add an extra ingredient for weighting links. As discussed in the previous chapter, this paper argues that link-based Pagerank algorithms do not generate an ideal "importance" ranking of pages. It is suggested that comparison of the contents of linked pages should be used to determine the weight of the link in the ranking equation. Using content analysis in weighting links is here considered to be a complementary factor to the standard link-based Pagerank computation.

$$R'(u) = c \sum_{v \in Bu} \frac{(1\text{-}d)(R'(v)) + d(M(v)\ R'(v))}{Nv} + (1 - c)E(u) \qquad (3.1)$$

Let M be a function returning the percentage match in content of pages v and u ($0 <= M(v) <= 1$). Let d be the content factor - the weight of the content measure in the equation - usually set to 0.1. In other words, certain percentage of the equation is weighted by the Pagerank value of the page, and the rest (usually 10%) is weighted by multiplying the content-match and the Pagerank.

3.2.1 Implementing Content-Sensitive Pagerank

Implementing content-sensitive Pagerank requires comparing pages' content during computation. This is done by using effective measurements, such as TFIDF, to create a term-vector for every page. The term-vector of a page contains the terms which have the discriminative characteristics of that page. Computing term-vectors for pages allows the differentiation between their contents.

The comparison function used in the algorithm computes the inner product of two term-vectors. If this is high the content of the corresponding pages tends to be similar. The function returns a match value ($0 <= d <= 1$) that reflects the content match of two pages. Normalization is done by giving the highest score (1) for pages with more than a 0.75 match. These pages are considered to contain similar content.

The content match for pairs of pages **is computed <u>once</u> for every link in the Web-graph** by calculating the inner product of the two pages. The content-match values are stored in memory, attached to their outlinks.

The computing of content-match between pages is done in the first distribution of Pageranks. Every page attaches its term-vector to all update messages sent for all of its out-linked pages. The out-linked pages, upon receiving a term-vector attached to a Pagerank update-message, execute the content-match function, and return the match value to the sender. The sender page store the

content-match value attached to its outlink object.

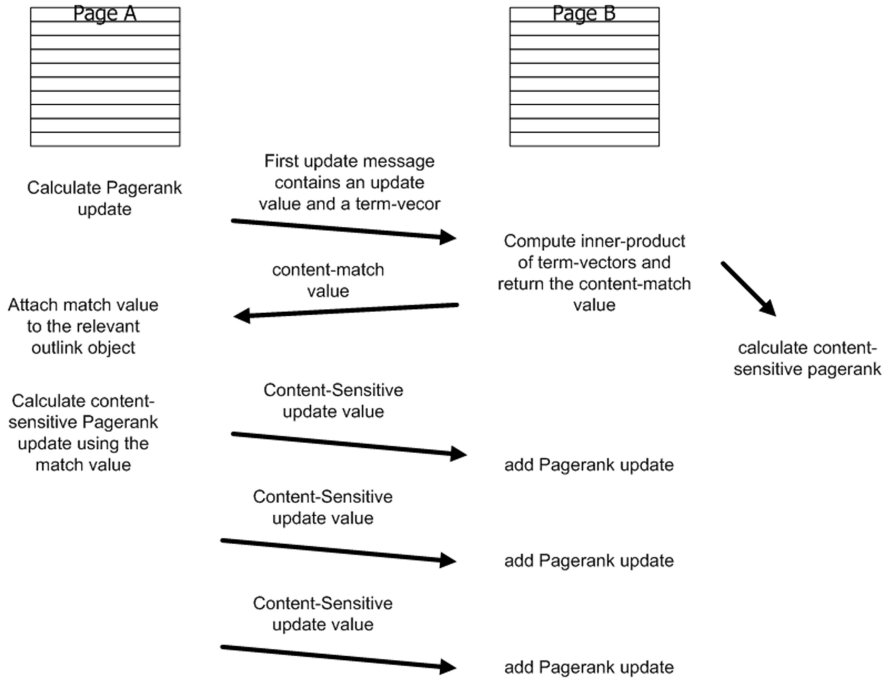

Figure 3.7: Content-Sensitive messaging protocol (Page A out-linked to page B)

At the end of the first distribution of Pageranks, every page in the system contains all the match-values of its outlinks. The content-match values are combined in the Pagerank equation as the next update messages generated by every page are multiplied by the content-match value separately for every outlink. Figure 3.7 shows the protocol of content-sensitive messaging.

3.2.2 Content Sensitive Pagerank: Performance Issues

Previous implementations incorporating content analysis in Pagerank computation reported remarkable levels of search accuracy (these results were precise in terms of their relevance to the query). The major drawback, common to all previous implementations, is the poor time performance achieved, which questions the viability of using such algorithms.

This project introduces procedures that incorporate content analysis into a distributed Pagerank implementation while actually improving time performance.

The content-sensitive protocol explained in the previous sub-section does not suffers from long computation time. The overhead involved in comparing pages' content is reduced to a single message sent for each link in the Web-graph. After the first distribution cycle update messages are calculated using the content-match value received by the first messaging cycle.

The content-sensitive Pagerank algorithm achieves immense improvement in performance as compared to the link-based Pagerank equation, in two important respects:
1. The time taken for the equation to converge.
2. The average number of Pagerank update-messages sent per page.

Explanations for these improvements are given in the experiments and conclusions chapters.

3.2.3 Content Reloading

The contents of large numbers of the pages in the Web frequently change, and the dynamic behavior of the Web should be reflected in any high quality ranking algorithm. The content-sensitive Pagerank system can be configured to reload the content of pages ay a pre-defined frequency. The frequency of reloading may be defined separately for web-servers according to their dynamic behavior. For example, news web-sites, such as Guardian.co.uk, tend to change the content of their pages more frequently than other web-sites. The content of their pages should therefore be reloaded more frequently. The reloading of content in pages will result in further distribution of Pagerank. In this way, a reduction in the content-match of two pages will cause a reduction in the weight of the link connecting them (in terms of the Pagerank update message, the update would be negative). However, the scope of this project did not permit this suggestion to be tested.

3.2.4 Statistics

The statistics of the Pagerank algorithm performance are gathered periodically by threads running in background, which analyze the Pages' database and take snapshots of pages information. The information maintained by every Page object is listed below:
- Pagerank value
- Number of update-messages sent
- Number of update-messages received
- Time of the last update-message sent
- Outlinks' URLs

- Content-match value for every outlink

The statistics are plotted periodically in two files:

1. A file containing the list of pages and their Pagerank values, ordered by Pagerank.
2. A file containing the information maintained in each page.

The convergence status, the number of messages sent/received in total and the number of messages sent/received on average are presented. An example of the statistics saved for a page is shown in below. The information represented includes the page's URL, the number of update-messages sent, the number of update-messages received a list of the outlinks' content-match values and the page's URLs.

```
http://education.guardian.co.uk/schools/
pagerank: 5.2565002
rcvmsg: 4479
sntmsg: 532
0.43 http://education.guardian.co.uk/higher/
0.44 http://education.guardian.co.uk/higher/careers/
0.42 http://education.guardian.co.uk/further/
0.47 http://education.guardian.co.uk/higher/research/
0.21 http://education.guardian.co.uk/Guardian/0,6961,,00.html
0.40 http://education.guardian.co.uk/higher/links/
0.43 http://education.guardian.co.uk/schools/favouritelesson/0,12186,1078472,00.html
1.0  http://education.guardian.co.uk/schools/specialreports/
0.48 http://education.guardian.co.uk/higher/comment/
0.40 http://education.guardian.co.uk/higher/worldwide/
1.0  http://education.guardian.co.uk/
```

3.3 Computing the Inner Product of Term-Vectors

As previously explained, the content-match of two pages is determined by computing the inner-product of their term-vectors. If the contents of the pages are similar, their corresponding inner-product tends to be high.

The design of content-comparison mechanism must give answers to the following questions: What exactly is a term? Which terms are to be included in a page's term-vector? How are terms' weights to be computed?

In this project the answers given to these questions are:

Terms The terms of an HTML page are sequences of non-space characters found by filtering and normalizing the page's term candidates. A page's term candidates are filtered by a lexicon (discussed below). Ideally, it should also be normalized by conversion to lower case and application of the Porter stemming algorithm‘[Por80]; due to lack of resources, this normalization is not implemented.

Lexicon The lexicon used for the term vector filtering is built by creating a sequence of term-frequency pairs (index) collected from a large pool of pages. As suggested in the "Term Vector Database" paper [KBS00], the first and the last thirds of the index should be excluded from the lexicon. The reason for eliminating the most frequent third is standard: such terms provide little discrimination across vectors [KBS00]. The least frequent third is eliminated because these terms are noisy and do not provide a good basis for measuring semantic similarity.

Term selection Selecting terms for inclusion in a page's term-vector is done using TFIDF methodology [SS95]: weighting a term in a page by dividing its appearance frequency in the page by the number of times it appears in the collection of pages.

Term Vector A sequence of term-weight pairs found in a page. The term-vectors used in this project are constructed only by the "important" terms, that is the terms which score high in the TFIDF measurement.

3.3.1 Building a Term-Lexicon

Designing a term-lexicon is a complex task. The best thing is simply to get hold of one. Commercial companies (such as Google and Yahoo) are using term-lexicons for indexing pages and for searching mechanisms. Using an index generated by a huge collection of textual sources (millions of pages) would be the best solution. Alternatively, one can build a term-lexicon by downloading good textual sources and then counting and indexing all the terms that appear in these sources. A good textual source may be a collection of pages which represents diverse subjects and themes.

Having failed to get hold of a term-index, this project introduces an implementation that builds a term-lexicon. The design of the term-lexicon generator is explained in the flow-chart diagram in Figure 3.8.

The implementation is done by processing thousands of web pages containing a wide range of subjects. More than 35,000 pages from the web-sites below where processed: 1. The Open-Directory-Project [ODP]: 15,700 pages
2. The Yahoo directory [YDi]: 14,950 pages
3. D-Lib Magazine (electronic magazine) [D-L]: 5,050 pages

In order to generate a balanced index, i.e. in order to construct the index from a content-diverse textual source, equal numbers of pages are processed for every subject introduced in the Directories and the Magazine.

A term-index of 225,000 different terms was created as a basis for a term-lexicon. The normalization process of selecting parts of the terms-index to create an effective term-lexicon is described in the Experiments chapter. The normalization process is done by eliminating terms that have little discrimination across a vector from the term-index.

The process of crawling, downloading and processing HTML pages is done by customizing an open-source crawler [Cra] and an open-source HTML-parser [Par]. The design, as described in figure 3.8, is rather conservative: backup files are used during the crawl and download processes in order to preserve the processed data.

Languages

Note that the textual source is mainly contains terms in the English language. Due to time limitations, this project does not provide solutions for handling different languages in the content-sensitive Pagerank algorithm.

"Stop" Term-List

A list of terms, called the stop-list, which has no discrimination characteristics over term-vectors (such as pronouns, prepositions, conjunctions, common adjectives and common verbs), are eliminated from the term-lexicon.

3.3.2 Term-Frequency versus TFIDF

The efficiency of the TFIDF measurement implementation is tested by comparing it to a frequency-comparison measurement. Implementing the two measurement approaches required storing the frequency and the TFIDF-weight for every term of the term-vectors.

Calculating the frequency of terms within HTML pages is done by counting the number of times the term appears either outside HTML tags or inside

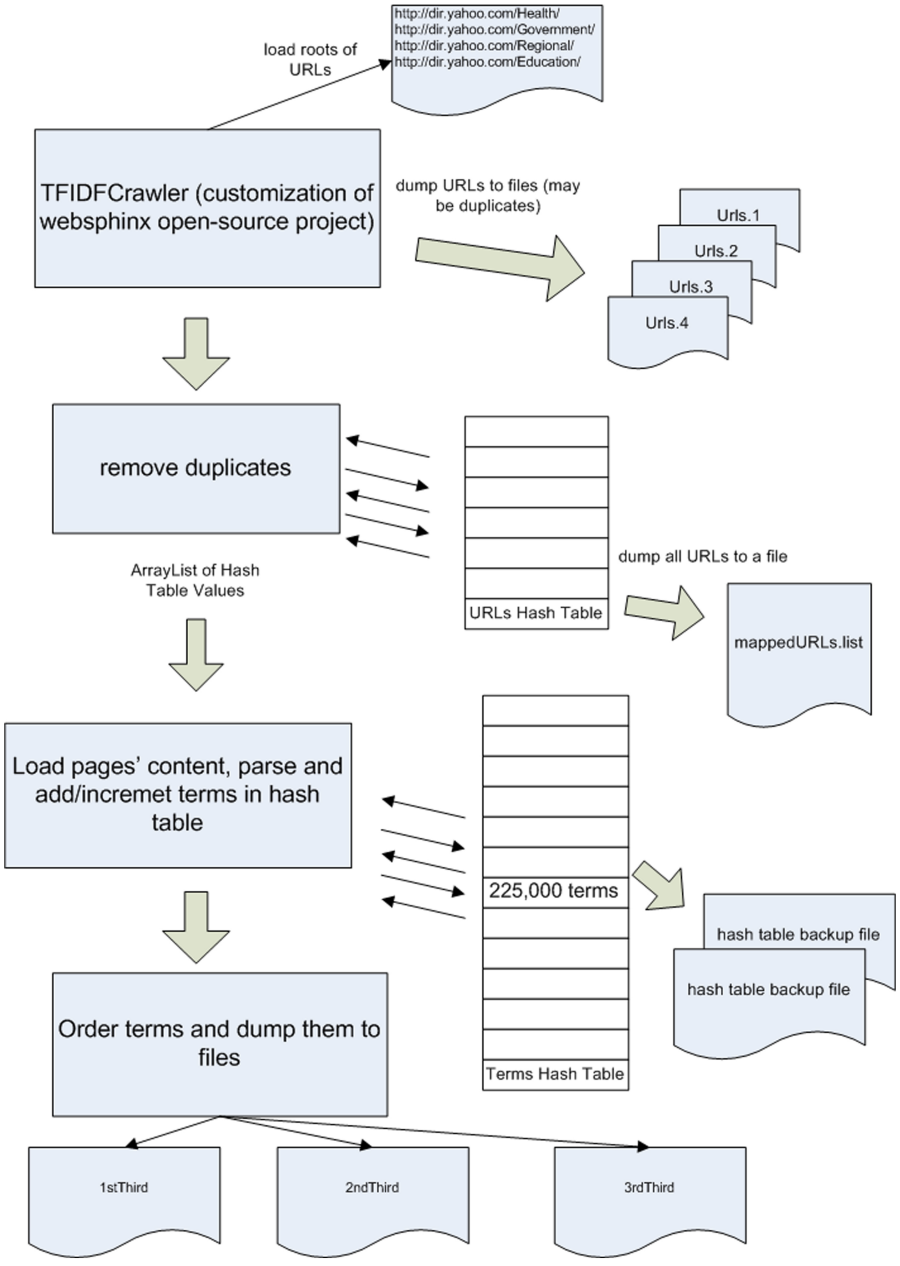

Figure 3.8: Generating a Term-Lexicon

HTML meta tags excluding terms found between scripts tags. The TFIDF measurement implementation is done by dividing the frequency of terms appearing in a page with their frequency retrieved by the term-lexicon.

Computing the inner product of term-vectors is done by comparing their terms and dividing the match results by the numerical magnitude of the term-vectors. Normalizing the comparison of term-vectors, a use of logarithmic scale is implemented in order to improve differentiation in comparisons results. Comparisons of term-vectors resulting in match percentages higher than 75% are rounded to 100% match. This is done to strengthen the links between pages that share a common theme.

3.4 Generating Web-graphs

Creating Web-graphs that represent the link-structure of a collection of web pages is not a trivial task. Many considerations have to be taken into account when creating middle-sized web-graphs for experiments. Some of these are detailed below:

- Inter-connectivity: pages should have outlinks to other pages within the Web-graph. Crawling too deeply will result in low linkage connectivity, whereas limiting the depth may result in a link structure which is disproportionate to the natural structure of the Web.
- Balance: crawling small web-graphs normally results in a collection of pages that are all linked to a small group of pages. These pages will then become a disproportionate source of rank that would adversely affect the propagation of Pagerank across the graph.
- Variety of themes: in order to test the content-sensitive algorithm, one must have a collection of pages that is content-diverse.
- A certain number of links between thematically unrelated pages is required to test the content-sensitive algorithm's effectiveness in reducing Pagerank for pages that are linked to others solely for the purpose of boosting their Pagerank.
- Web-sites with many dynamic links (such as asp, jsp, php) may create major difficulties when testing successive executions of the content-sensitive algorithm. Measuring different configurations of the algorithm on the same data-set requires a less dynamic environment. The BBC web-site (http://www.bbc.com/) is one example of a dynamic web-site.
- Size: The web-graph must be big enough to represent a true view of the Web, but not too big to prevent the performance of complex tests.
- domain names: diversity of host names of URLs is needed for demonstrating the structure of websites in the Web.
- Inter-connectivity between pages that belong to the same host: a large part of the linkage in the Web is within the websites domain. This is necessary in order to have a proportional number of local linkage between pages (local outlinks of a page triggers a function call in the distributed algorithm, instead of an RMI message).

All of the constraints above that need to be taken into account when creating Web-graphs required a careful examination of many generated Web-graph samples.
Suitable Web-graph were generated from the root URL of the Guardian's Website (see figures 3.9 and 4.11).

Size of Web-graph	2K
root	http://www.guardian.co.uk/
crawler search method	depth first
crawler max depth	30
number of outlinks in total	46,280
average outlinks per page	21
number of cul-de-sac pages	5

Figure 3.9: Standard Web-Graph used in most experiments

Size of Web-graph	6K
root	http://www.guardian.co.uk/
crawler search method	depth first
crawler max depth	30
number of outlinks in total	154,559
average outlinks per page	24
number of cul-de-sac pages	13

Figure 3.10: Standard Web-Graph used in performance experiments

3.4.1 Web-Graphs Representation

Web-graphs are generated as files which represent the link structure the crawled area of the Web. The representation of the Web-graph is determined by the needs of the distributed Pagerank algorithm. For example, synchronous algorithms would normally require information on the number and sources of the inbound-links of a page, whereas the non-iterative asynchronous algorithm proposed in this project requires information on the outbound-links' URLs of a page.

3.4.2 Outbound-links Exclusion

Not all links found in HTML pages are useful in the Pageank equation. The following links are excluded from the generated Web-graphs:

- Duplicates
- Non-Hypertext links: links to objects other than HTML pages (e.g. pictures, icons, flash objects)
- Links to pages which are not included in the Web-graph
- Self pointing links: tends of boosting the Pagerank values of popular pages out of proportion
- Dynamic links (such as asp, aspx, jsp, php): are not pointing to static pages

- Email links
- Pointers to a location within a page (using # in the URL address) are removed. For example, the URL http://www.whitehouse.gov/news/index.html#2004 is reduced to http://www.whitehouse.gov/news/index.html

Chapter 4

Experiments

This chapter describes experiments carried out on both the distributed asynchronous Pagerank algorithm and on the content-sensitive algorithm developed in this project. These experiments aim to explore various aspects of the two algorithms, and also to test the quality of the ranking system developed.

4.1 Testing Google's Crawling Cycles

This test examines the caching frequency of certain types of pages in the Google search engine. By examining the Google's caching mechanism one can learn about the scalability problem in centralized search engines, i.e. the challenge of coping with the scale of the Web and with its dynamic characteristics.

URL	update	update
http://www.doc.ic.ac.uk/∼sm1603	25.6.04	30.8.04
http://www-lp.doc.ic.ac.uk	2.7.04	-
http://www.doc.ic.ac.uk	5.8.04	22.8.04
http://www.ic.ac.uk	19.8.04	23.8.04
http://www.yahoo.com	19.8.04	20.8.04
http://www.CNN.com	19.8.04	20.8.04
http://www.nba.com	11.8.04	24.8.04

Figure 4.1: Google's caching cycles

Figure 4.1 shows the update cycles for various URLs. It shows that "important" URLs, such as CNN.com and yahoo.com, are cached on a daily basis, whereas less "important" pages, such as www.doc.ic.ac.uk/∼ sm1603/, are updated infrequently.

It is clear that Google uses different parameters (such as start points, depth and width of search, and frequency) for crawling different areas of the Web. By

doing this, Google can achieve a wider view of the web, and, for some predefined web-sites, an up-to-date view as well.

As mentioned in chapter 2, it is estimated that Google's Pagerank computation is carried out once every 5 weeks. Taking into account the fact that some parts of the web are crawled less frequently than this, one can conclude that part of the matrix used for Pagerank calculation, which represents the link-structure of the Web-graph, is based on non-updated information.

The fact that some web-sites are crawled much more frequently than others raises some questions:

- What criteria determine how frequently Google's crawlers visit particular pages?
- Is the crawling frequency of particular areas decided by an automatic process or by human decision?
- How does the claim that the PageRank algorithm "relies on the uniquely democratic nature of the web" [Goo] fit with the fact that some web-sites are crawled more frequently than others?

4.2 Testing Distributed Asynchronous Link-Based Algorithm

The next sub-sections describe the tests carried out on the link-based distributed asynchronous algorithm developed in this project.

The quality of an algorithm can be measured in many ways. The following sections discuss some of them:

- ranking stability: examine variations in Pagerank
- convergence performance: time and communication load
- convergence behavior: progress and accuracy over time

All tests were done in a cluster of several Linux machines, each equipped with 64M average RAM, connected by LAN network. The epsilon value used in most tests is 10^{-4}.

4.2.1 Quality and Accuracy: Variations in Rank Results

Ideally, the quality of the asynchronous link-based Pagerank algorithm, in terms of ranking value per page, should be tested in comparison with a synchronous centralized algorithm. As the aim of the project is to examine the content aspects of a distributed Pagerank algorithm, the Pagerank quality is measured against that of a centralized version of the same algorithm, run by executing the algorithm on a single machine.

These tests aims to measure the influence of network latency, congestion, failures and other parameters that evolve from the natural structure of every network.

Running successive executions of the system on a single machine, using the same number of threads in the first test, and different numbers in the second, show variations in Pagerank results (figures 4.2 and 4.3). In other words, the asynchronous Pagerank algorithm resulted in slightly different ranking values when running successive executions. Figure 4.4 shows variations in Pagerank values when running the algorithm on 5 machines compared with a single machine execution. On the other hand, the ratio, which represents the relative difference in Pagerank, is very close to 1 for all pages, indicating very little difference between the two. The absolute difference for each page (ABS(firstPR - secondPR)) is close to 0 for all pages.

These variations in Pagerank values did not appear to be significant as there are **no significant changes in the relative rank**. Absolute differences in

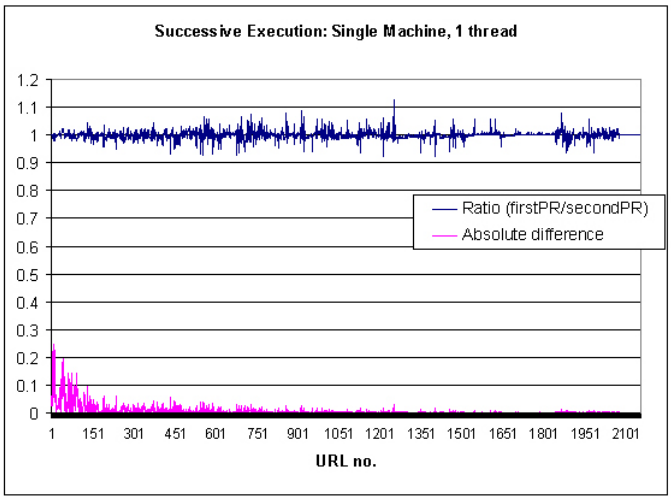

Figure 4.2: Variations in Pagerank results: successive executions on a single machine, 2k Web-graph, ordered by pages' Pagerank

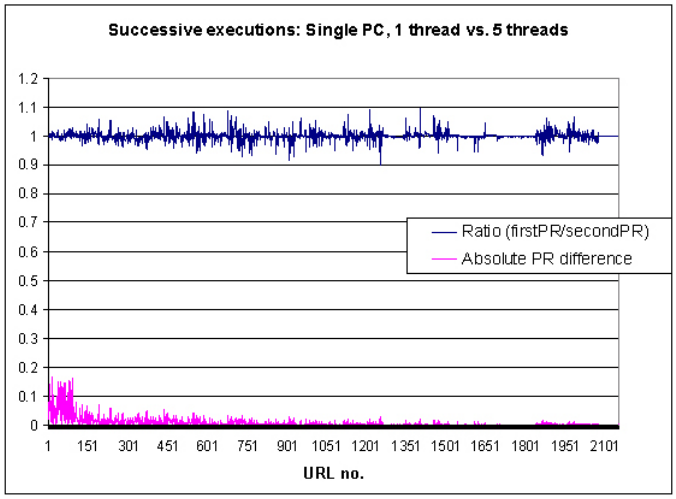

Figure 4.3: Variations in Pagerank: single machine, 1 thread vs. 5 threads, 2k Web-graph, ordered by pages' Pagerank

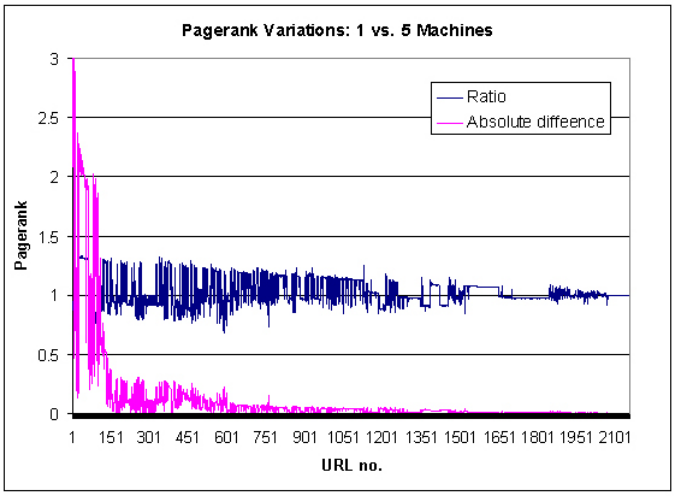

Figure 4.4: Variations in Pagerank: 1 vs. 5 machines, each running on 5 threads, 2k Web-graph

Pagerank are nominally larger in the high scoring pages, but the ratio remains stable, indicating that relative rank is maintained.

The graphs displayed here and figure 4.5 show that the asynchronous algorithm suffers from little variation in Pagerank values. The behavior of the algorithm in a large scale Web-graphs cannot be determined from these experiments however. Due to the relatively small scale of the Web-graph used, these experiments can only indicate the exitance of variations, which are caused in part by the natural characteristics of the network and in part by the characteristics of the non-iterative asynchronous algorithm.

compare	average ratio	average absolute diff.
one machine: successive	0.999	0.007
one thread vs. 5 threads	1.001	0.008
one PC vs. 5 PC	1.017	0.113

Figure 4.5: Variations in Pagerank: average ratio and average of absolute difference

An important feature of the algorithm revealed in these tests is that relative rank is maintained over the entire collection of pages, as reflected in the ratio value obtained in all experiments.

An experiment aiming to reduce the variations of results using a smaller

epsilon value is introduced in figure 4.6. The results show no improvement in accuracy of Pagerank when using smaller epsilon value (the relative error threshold is computed by (PRd - PRc)/PRc). As demonstrated, the use of a smaller epsilon dramatically increases communication overhead and convergence time.

epsilon value	avg relative error threshold (Pd - Pc)/Pc	average ratio	avg messages sent per page	convergence time (5 machines)
0.01	0.0922	1.0180	53.3	214 sec
0.005	0.0936	1.0211	121.7	394 sec
0.001	0.0920	1.0183	557.3	1664 sec

Figure 4.6: Pagerank accuracy: comparing centralized and distributed (5 machines) executions of 2K Web-graph using different epsilon values

4.2.2 Convergence: Progress and General Performance

This experiment shows that the distributed link-based Pagerank algorithm achieves a good approximation of the final Pagerank values in just over half the total computation time. The table in figure 4.7 displays the progress of Pagerank accumulation. Note that 98.38% of the total Pageank was accumulated after only 56% of the computation time.

SnapShot (seconds)	average accuracy % of final results	average error-threshold: (PR-finalPR)/finalPR	% of pages converged	average update-messages sent per page
30	77.86	0.2213	6.98	28.56
60	89.19	0.1080	23.62	41.5
90	97.06	0.0293	41.75	49.39
120	98.38	0.0016	61.23	52.12
214 (converged)	100.00	0.0000	100.00	53.3

Figure 4.7: Pagerank equation progress over time (2k Web-graph, using 10 machines)

Interesting algorithm characteristic is revealed by comparing the second and fourth columns: although most of the Pageranks accumulated very quickly (second column), the progress of pages that reached the convergence state is roughly

linear (fourth column).

This characteristic also reflects the quality of the Web-graph used in the experiment. Having a balanced Web-graph (i.e. a collection of Web pages that are nearly equally interconnected) is important for achieving reliable results.

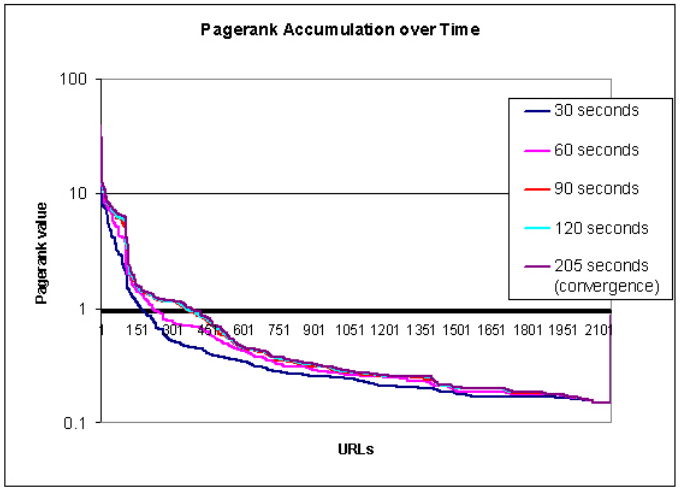

Figure 4.8: Pagerank Progress: taking snapshots of the equation

4.3 Tuning the Content-Sensitive Algorithm

The following experiments introduce several methodologies and configurations for comparing the content of pages. Different methods of comparison are used to achieve different search goals. Based on the positive results reported in the "Term Vector Database" paper [KBS00], this project attempts to build a Term-Lexicon for improving the use of TFIDF as a measure of comparison. The next sections describe tests on various ways of comparing content.

4.3.1 Constructing Term-Lexicon for Filtering Term-Vectors

As explained in the Architecture chapter, Term-index is a list of term-frequency pairs that appear in a wide collection of Web-pages. The Term-index was generated by processing 35,000 pages and creating a vector of 225,000 terms. Following Bharat, Maghoul, and Stata report [KBS00], the first and last thirds of the Term-index collected (the most and least frequent terms) should be be omitted when creating a lexicon (Term-lexicon). The is because these terms provide little discrimination across pages.

The following experiments aim to achieve similar results - a lexicon able to filter out unnecessary terms - using a smaller resource to create the Term-lexicon. The quality of the lexicon is tested by comparing pages thought to have a particular degree of content relation, measuring the content-match results achieved in different configurations of the lexicon.

Six different categories indicating degrees of content-relation between pages are suggested:

- Identical (comparing a page to itself): 100% content-match.
- 6 key-words query in Google: similar content of pages.
- 4 key-words query in Google: a high content-based match between pages.
- 2 key-words query in Google: some degree of relation between pages.
- 1 key-word query in Google: low content match between pages.
- Randomly chosen pages: thematically unrelated pages.

10 pairs of pages from each category are chosen for comparison (the pages' URLs are listed in the appendix). Using TFIDF and frequency comparison measures, various configurations of the lexicon are tested with the aim of finding a balanced division of the Term-index that will enable it to achieve a reliable content-based comparison.

Figures 4.9, 4.10 and 4.11, show the results of comparing pages using different divisions of the Terms-index made.

Experiments have shown that using the middle third of the index, as suggested by Bharat, Maghoul, and Stata, is not sufficient to determine the quality of the content-match comparison. This procedure is not suitable for an index built out of relatively small textual sources, as shown in figures 4.9 and 4.10.

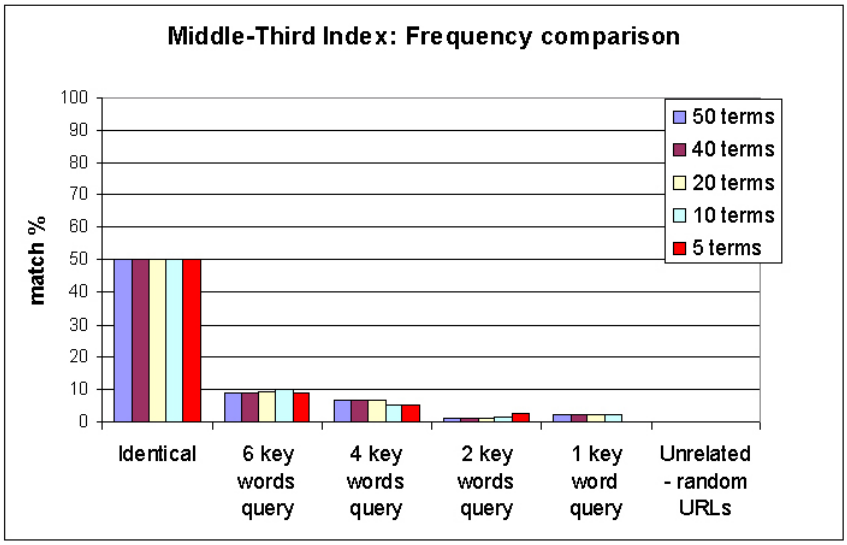

Figure 4.9: Term-index division: middle third, Frequency

These results led to further experiments using divisions of the Term-index in which the percentage of the lexicon created out of the complete index is higher.

Dividing the Term-index into 10-60-30 % (creating an index of the middle 60%), 20-50-30%, 5-55-40%, 5-65-30%, 5-75-20% and 100% did not adequately improve the comparison results. For example, see figures 4.11 and 4.12.

The best results were attained using a Term-lexicon of 90-10%: 90% corresponding to the most frequent terms, 10% corresponding to the least frequent terms. This configuration does not exclude the most-frequent terms from the index, as their weight becomes insignificant in the TFIDF measurement (as will be explained in the following section). Figures 4.13 and 4.14 demonstrate the excellent comparison results using both TFIDF and frequency measurements.

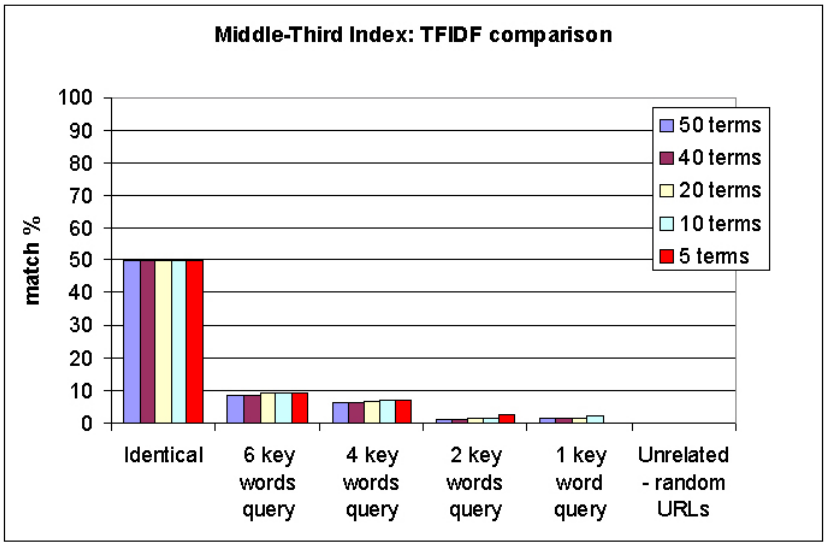

Figure 4.10: Term-index division: middle third, TFIDF

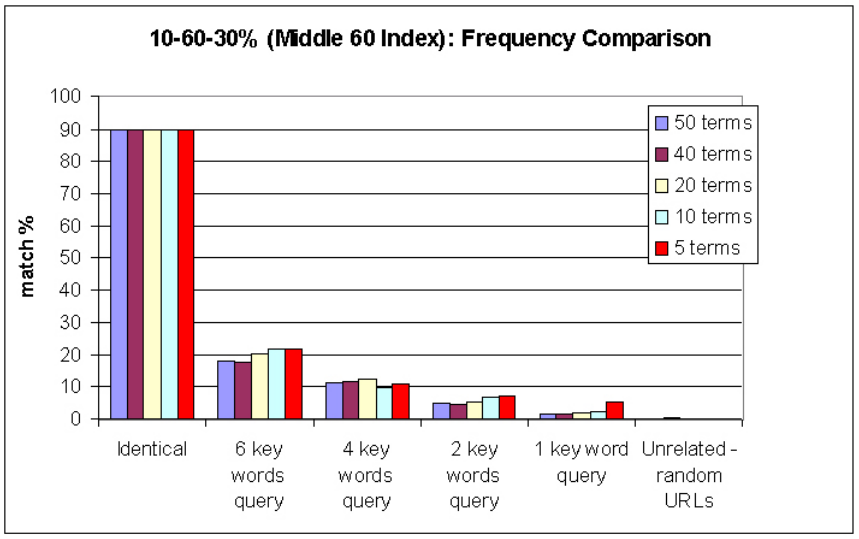

Figure 4.11: Term-Vector division: 10-60-30% (creating an index of the middle 60%).

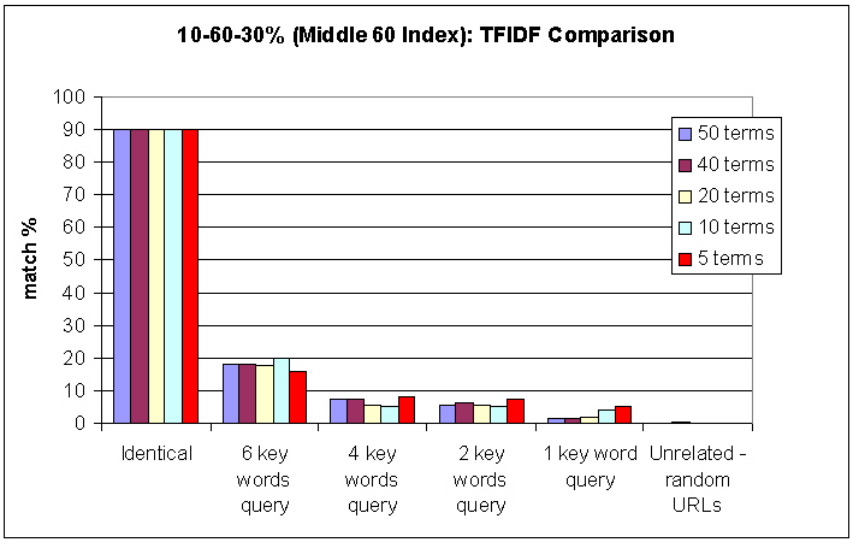

Figure 4.12: Term-index division: 10-60-30% (creating a lexicon out of the middle 60%).

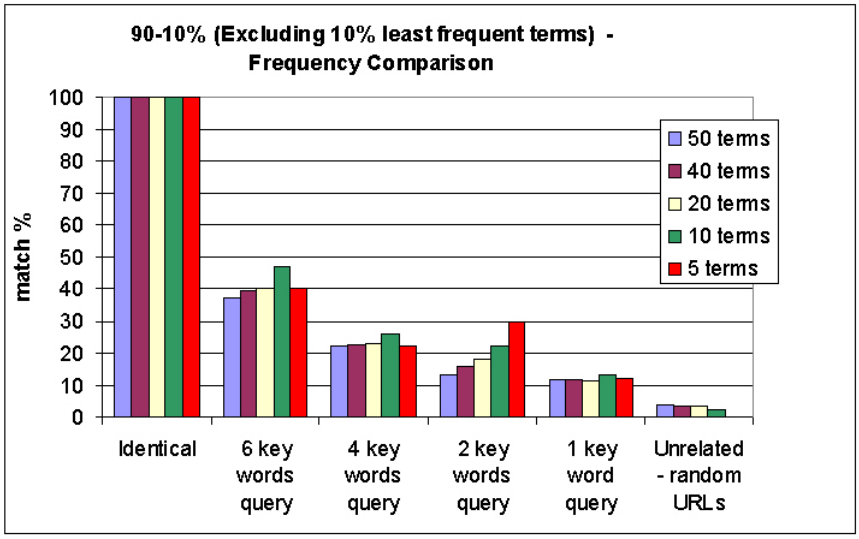

Figure 4.13: Term-index division: 90-10% (excluding the 10% least frequent terms from the lexicon).

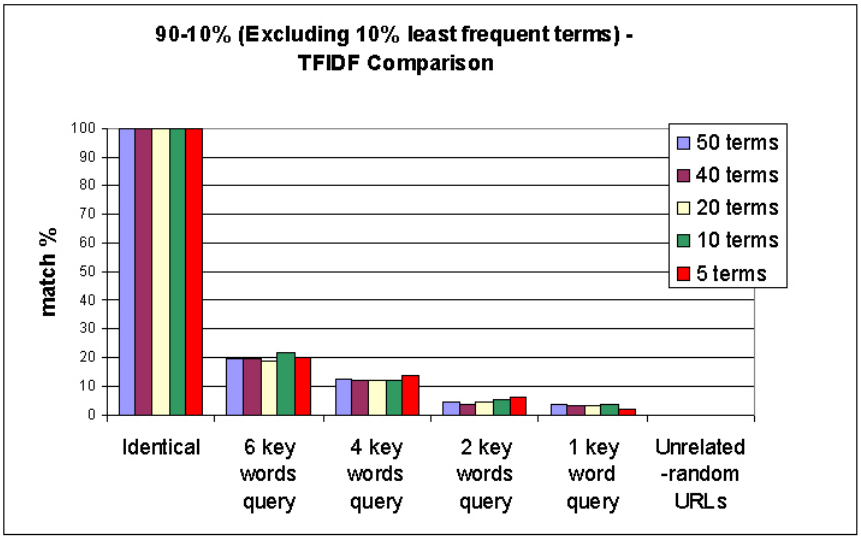

Figure 4.14: Term-index division: 90-10% (excluding the 10% least frequent terms from the lexicon).

4.3.2 Frequency versus TFIDF Measurements

Using simple frequency measurement in content comparison (the inner-product of pages' most frequent terms) may lead to inaccurate content-match results. As can be seen from figures 4.15 and 4.16, the match percentage of unrelated pages is relatively high, when it should be zero.

The use of the frequency measurement normally scores high in comparison to the TFIDF, as terms appearing frequently in the Term-lexicon are not omitted from the comparison.
Thus, this paper suggests that the TFIDF achieves more accurate results when compared to the frequency measurement.

4.3.3 Characteristics of Content-Sensitive Ranking

TFIDF is used for the comparison of pages' contents, as explained in chapter 4, in two phases:
1. Classifying pages: building a term-vector for a page by choosing terms relatively important for the page's theme (terms which score high in the TFIDF measurement). These terms, therefore, can be used as classifying characters for their pages.
2. Comparing term-vectors: defining the percentage correlation between pages'

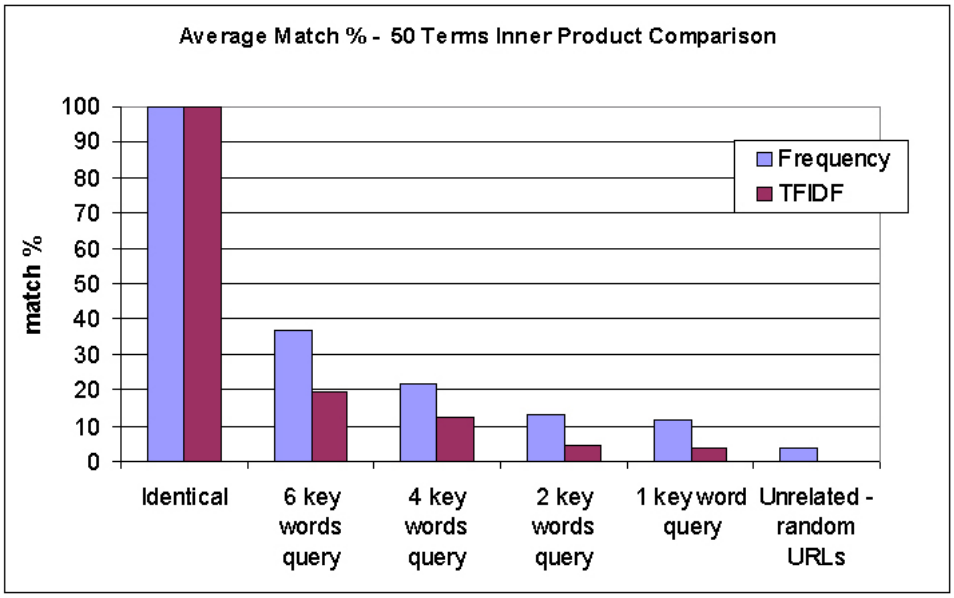

Figure 4.15: Frequency vs. TFIDF: 50 terms comparison.

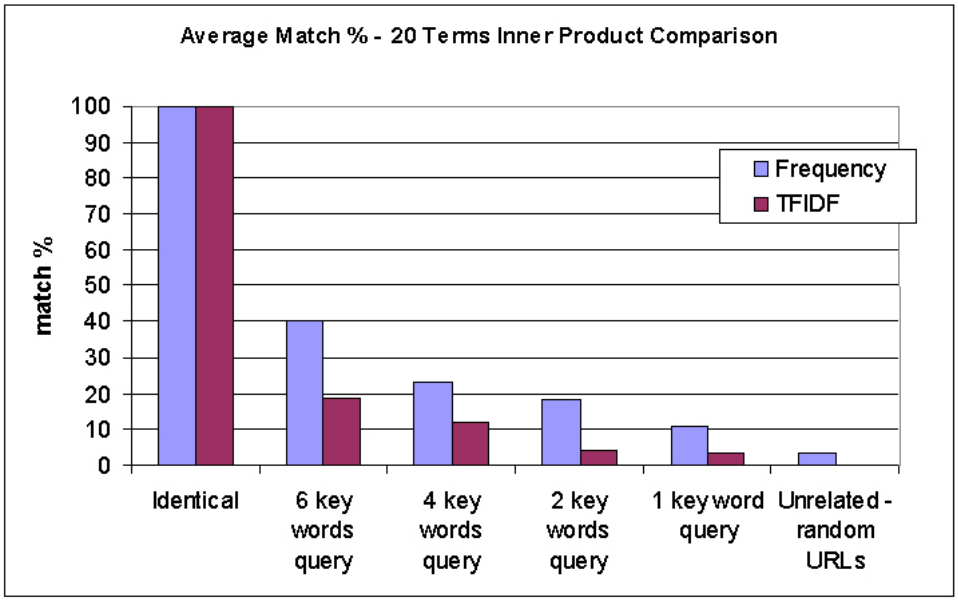

Figure 4.16: Frequency vs. TFIDF: 20 terms comparison.

contents.

It is reasonable to assume that when comparing large size of term-vectors the content match will be "wider". In other words, characterizing pages using a wider selection of "important" terms will relate not only to the specific content of the page, but also to the further context to which the page relates.

The number of terms used for comparing pages in the Content-Sensitive Pagerank algorithm is an important factor for defining its ranking characteristics. Using small-size term-vectors in TFIDF-based comparison causes small grouping of pages, as the distribution of Pagerank values in the equation is to pages which are specifically related to each other. On the other hand, the use of bigger term-vectors brings about grouping of pages according to a wider content relation.

This paper suggests that calculating content-sensitive Pagerank using different sizes of term-vectors can be used for various types of search. Using small-size term-vectors in Pagerank computation provides a better ranking scheme for a specific search, while using bigger-size term-vectors is preferable for a more general search.

For example, a comparison of Oxford's and Cambridge's Computing Science Departments' main pages, which are considered relatively "important" and thematically related, gave different results when using a different number of terms. When using a 10-terms-TFIDF measure the content match between the two pages was relatively low, while when using a 20-terms-TFIDF measure the content match was higher (see figure 4.17).

method	match percentage
10 terms TFIDF comparison	10%
20 terms TFIDF comparison	25%

Figure 4.17: Comparing www.cl.cam.ac.uk and web.comlab.ox.ac.uk/

Examining the terms used in every comparison leads us to conclude that the words used for 10-TFIDF were very specific (such as the department's address, name of university, the head of the department), whereas the use of 20 terms enables a larger base of comparison.

4.3.4 Convergence Progress of the Content-Sensitive Algorithm

The table in figure 4.18 shows the progress of the Pagerank computation. The information was collected by taking snapshots of the system every 30 seconds. The Web-graph and the system configuration used both in this experiment and in the equivalent linked-based one (section 5.2.2) are similar.

SnapShot (seconds)	average accuracy % of final results	average error-threshold: (PR-finalPR)/finalPR	% of pages converged	average update-messages sent per page
30	81.69	0.1830	18.47	27.21
60	94.89	0.0510	39.15	39.50
90	98.62	0.0137	59.55	44.12
120	99.43	0.0056	75.68	45.25
192 (converged)	100.00	0.0000	100.00	45.94

Figure 4.18: Content-sensitive (TFIDF20) equation progress over time (2k Webgraph, using 10 machines)

As can be seen from the table in Figure 4.18, Pagerank converges to a reasonable tolerance after roughly 45 update messages sent per page. Almost all Pagerank values (99.43%) were accumulated in only 62% of the total equation time.

4.4 Link-Based Algorithm versus Content-Sensitive Algorithm

Figure 4.19 shows the results of the content-sensitive and the link-based algorithms. The reduction of Pageranks for pages pointed to by thematically unrelated pages is apparent.

A closer look at the filtering of the Pagerank contribution of content unrelated pages is provided in figure 4.20. Some particular examples of this behavior are discussed in the last section of this chapter.

Figure 4.21 summarizes the Pagerank computation performances of the link-based algorithm compared with the content-sensitive one. As can be seen from the table, the content-sensitive algorithm outperformed the link-based algorithm in every performance parameter. There are two main reasons for this:

1. Elimination of unjustified Pageranks : the propagation of Pageranks from pages linked to thematically unrelated pages is dramatically reduced in the computation.

2. The strengthening of "related" links (links connecting thematically related pages) and the weakening of "unrelated" links (links connecting unrelated pages) causes the grouping of pages into subject-related areas. Manipulating the propagation of Pageranks in this way influences the performance of the Pagerank computation as the concentration of Pageranks in the center of subject-related groups partitions the computation.

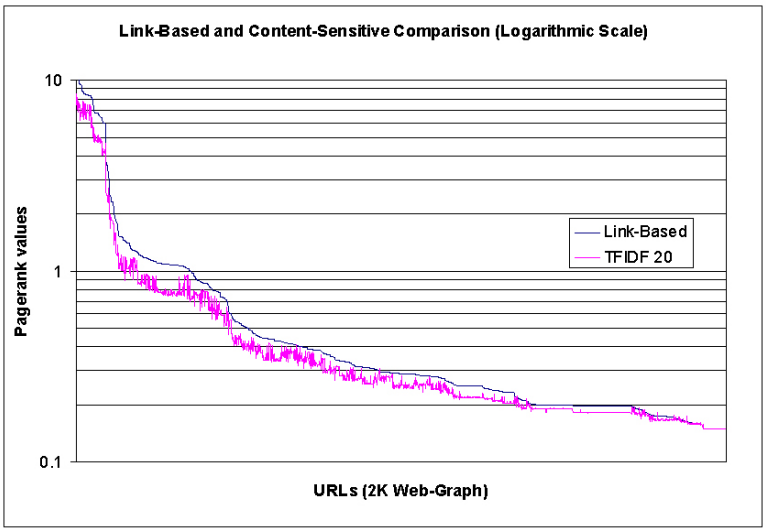

Figure 4.19: Pagerank results: link-Based compared with content-sensitive algorithms

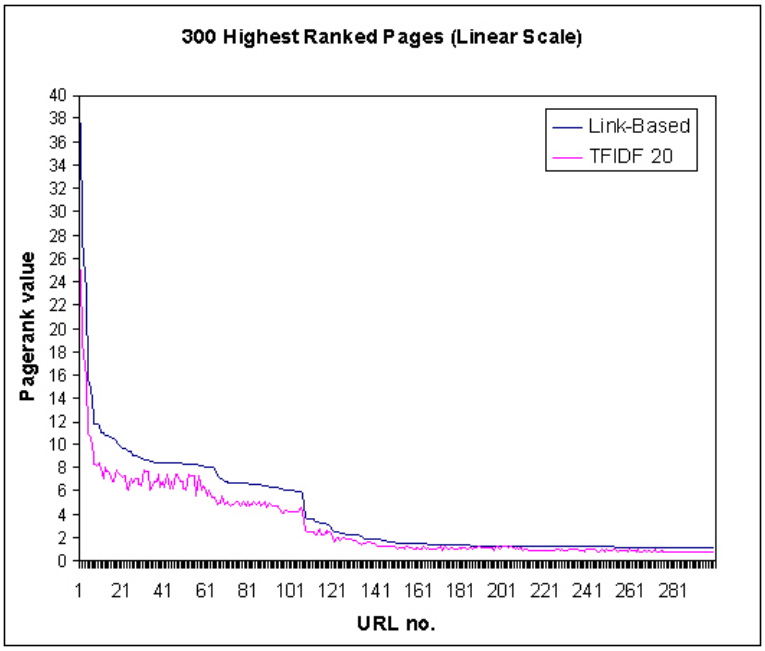

Figure 4.20: Pagerank variations in Content-Sensitive compared with Link-Based

SnapShot (seconds)		average accuracy % of final results		average error threshold (PR-fPR)/fPR)		% of pages converged		average update messages sent per page	
link	content	link	content	link	content	link	content	link	content
30	30	77.86	81.69	0.2213	0.1830	6.98	18.47	28.56	27.21
60	60	89.19	94.89	0.1080	0.0510	23.62	39.15	41.5	39.50
90	90	97.06	98.62	0.0293	0.0137	41.75	59.55	49.39	44.12
120	120	98.38	99.43	0.0016	0.0056	61.23	75.68	52.12	45.25
214	192	100.00	100.00	0.0000	0.0000	100.00	100.00	53.30	45.94

Figure 4.21: Comparing link-based and content-sensitive algorithms: equation performance

4.4.1 Reducing "Unjustified" Ranks: Boosting Pagerank Prevention

Installing links for the sole purpose of boosting a page's Pagerank is a technique commonly used by web-masters and programmers.
One of the motives for weighting links on the basis of content analysis is to avoid this corruption. By incorporating content analysis in Pagerank computation it is possible to diminish the influence of links that are not thematically related to their target pages.

The reduction of "unjustified" rank improves the quality of the ranking of pages. Using the words of Page and Brin, pages "voting" to other pages which are content related, has a higher influence of their ranks. The equation can be described as a "democratic" process of "experts". The result is that pages gain high ranks by being relevant to their topic.

The next experiment demonstrates a dramatic reduction of Pagerank for pages that are likely to be unrelated to any of the pages in the web-graph.

The graph in 4.22 shows the reduction in Pageranks of the 20 pages whose Pageranks were most reduced. The test was performed on a 1000 pages Web-graph generated from the root URL www.CNN.com/. The Web-graph is characterized by its relatively close-knit thematic relation, as part of the content is related to the CNN web-site. In such an environment, locating pages that are thematically unrelated is easier.

Looking at the list of pages, one can find pages that are written in different languages (see figure 4.23). As discussed in chapter 2, content analysis is problematic in multi-language environments, but for the purpose of this test these pages can be treated as thematically unrelated to the majority of pages on this Web-graph.

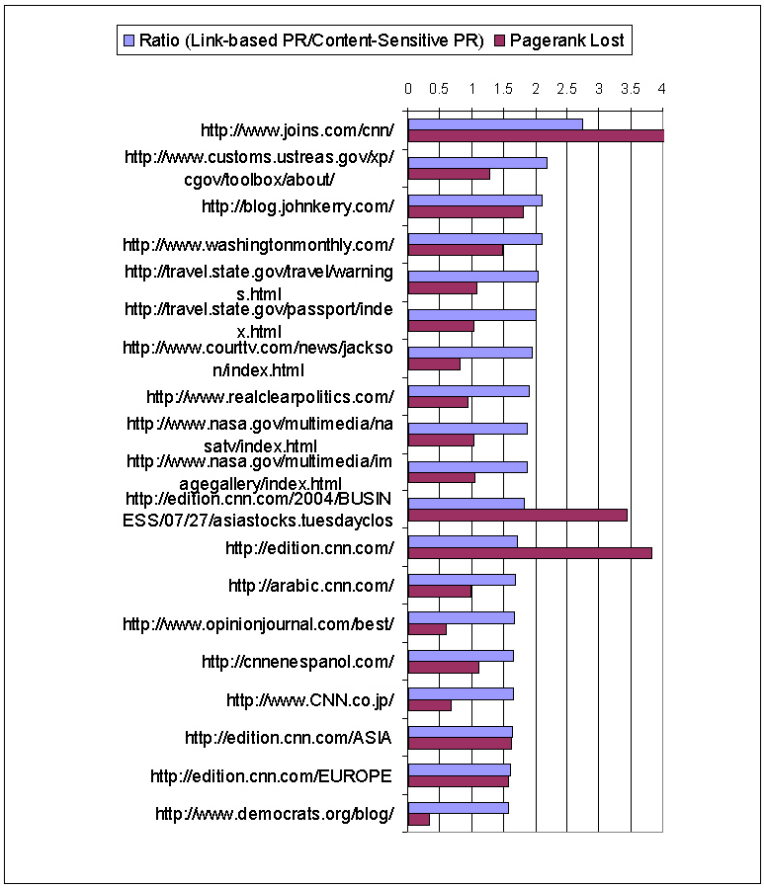

Figure 4.22: Relative reduce in Pagersnk: Top 20 pages of 1K CNN.com Web-Graph

URL	Language
http://www.joins.com/cnn/	Japanese
http://arabic.cnn.com/	Arabic
http://cnnenespanol.com/	Spanish
http://www.CNN.co.jp/	Japanese

Figure 4.23: Reduce in Pagersnk for pages written in diffeent languages

Other examples, in which it is harder to judge whether the page is unrelated thematically, are specified in the next figures:
- figure 4.24, less-related thematically.
- figure 4.25, popular pages (pages with many inbound-links) that are lacking in textual content.
- figure 4.26: very popular pages whose content is extremely general.

URL	Theme
http://travel.state.gov/travel/warnings.html	an official warning for US travellers
http://www.courttv.com/news/jackson/index.html	Michael Jackson in court TV
http://edition.cnn.com/2004/BUSINESS/07/27/asiastocks.tuesdayclose/index.html	Worlds-stoke markets Tuesday's summary
http://www.democrats.org/blog/	Democrats' Party blog

Figure 4.24: Reduce in Pagersnk: less-related thematically to the general collection

URL	Theme
http://www.cnn.com/video/videopage.html	video-clip links
http://www.nasa.gov/multimedia/imagegallery/index.html	NASA, Image gallery
http://www.nasa.gov/multimedia/nasatv/index.html	NASA TV, Index of programms

Figure 4.25: Reduce in Pagersnk: popular pages which are lack in textual content

URL	Theme
http://edition.cnn.com/	International edition main page
http://edition.cnn.com/ASIA	News from Asia main page
http://edition.cnn.com/EUROPE	News from Europe main page

Figure 4.26: Reduce in Pagersnk: popular pages with very general themes

Chapter 5

Conclusions

5.1 Contribution

The project puts forward a distributed algorithm that is appropriate for the Web's structure. The aim of introducing a distributed asynchronous non-iterative Pagerank algorithm was achieved, and a full implementation of the distributed system was developed and tested.

Previous distributed implementations of the Pagerank algorithm introduced in this paper rely on a certain degree of synchronization. This paper criticizes the use of any degree of synchronization in distributed Pagerank algorithms because, as explained in previous chapters, this makes them unsuitable for large scale networks. However, not using synchronization in a distributed Pagerank algorithm slightly reduces the accuracy of ranking results.

The testing of the distributed asynchronous algorithm showed a certain degree of variations in ranking results. These variations, however, could be argued to be insignificant, as pages still preserved their relative rank.

The behavior of the algorithm in a large scale Web-graph (millions of pages) cannot be estimated or concluded from the 2k Web-graphs experiments performed. The influence of latency, load, and other network characteristics on a small Web-graph is relatively higher than in a large Web-graph as the weight of every link in the Pagerank equation is higher, and hence changes in network behavior have more influence on the propagation of ranks. On the other hand, large networks are more vulnerable to network instability. Testing the distributed asynchronous Pagerank algorithm on a large scale Web-graph could show how significant these differences are.

The second aim of this project, to examine ways of incorporating content analysis into the Pagerank computation in order to improve its ranking qual-

ity, has also been achieved. An implementation of a distributed non-iterative content-sensitive Pagerank algorithm has been introduced. The algorithm presented combines comparison of pages' contents and link-structures in a distributed Pagerank computation. Taking advantage of the natural partitioning of pages to web-servers, the content-sensitive Pagerank implementation produced excellent results in both rank quality and general computation performance.

The project evaluated the two algorithms by comparing several aspects of performance and rank quality. The challenge of developing a content-sensitive algorithm that does not suffer from poor computation performance was met, as the content-sensitive algorithm outperformed the link-based one on all performance parameters.

Other major achievements and discoveries of this project are as follows:

- Various solutions for coping with the scalability and availability challenges of search engines have been put forward.
- Close approximation to final Pageranks for all pages in the collection was reached after just over half the total computation time.
- The strengthening of "related" links (links connecting thematically related pages) and the weakening of "unrelated" links (links connecting unrelated pages) in the content-sensitive algorithm, causes the grouping of pages into subject-related areas. This project showed that manipulating the propagation of Pageranks this way improves the equation performances of the Pagerank algorithm for two reasons: the reduction of "unjustified" Pageranks and the concentration of Pageranks in the center of subject-related groups partitions the computation.
- Computation performance shortcomings caused by the overhead from analyzing the content of pages was overcome by performing the content-comparison only **once** for every outlink, and by caching the content-match values attached to the outlinks objects.
- Reduction in "unjustified" Pageranks was demonstrated with the content-sensitive Pagerank algorithm. The decrease in Pageranks improves the quality of the overall ranking as well as restricts the possibilities for manipulating ranking results (spam).
- A methodology for customizing the ranking results by using different sizes of term-vectors in content-sensitive Pagerank was investigated.
- A caching mechanism for reducing communication overhead caused by DNS lookups was introduced.

5.2 Future Work

The project focused on ranking algorithms that are part of the architecture of search engines. The ranking results generated by the algorithms developed here were not tested with respect to quality of search results. To complete the research and achieve accurate assessment of the implementation in this matter, experiments with the algorithms in a large scale web-graph are required.

The project engaged three major Computing-Science fields: Information-Retrieval, Distributed-Systems, and Ranking-Algorithms. Hence, further research on the content-sensitive and distributed Pagerank algorithms can be conducted in several directions. There is considerable potential for further research, both in new areas and in the areas already touched on here.

- Testing the distributed non-iterative algorithm on a large scale web-graph: experiments conducted in this project showed slight variations in ranking results. A complementary test would be to examine this behavior on a large scale Web-graph (millions of pages).
- Testing of the immediate influence of changes in pages' contents was not performed in this project. An interesting experiment would be to examine, within a large scale web-graph, the influence of changes in the content of "important" pages on the ranking of other pages. Reflecting the dynamic behavior of the Web by using a real-time ranking system would be a good solution to the need for an up-to-date search mechanism.
- Using TFIDF, or any other methodology, for comparing content of pages is argued to be effective for most cases. However, there are some cases where the content comparison is not sufficient to determine content match. For example, pages containing no textual information, pages containing similar themes but written in different languages, and pages describing similar content in different vocabulary cannot be reflected in the measurement. Thus, further research for optimizing the comparison of content between pages is required.
- This project has not offered solutions for preventing spam (manipulations of ranking results) for the suggested content-sensitive algorithm. Further research of the drawbacks of the content-sensitive algorithms would be desirable.

Appendix A

Experiments and Design Diagrams

A.1 Inner-Product Comparison

The Pages selected for the comparison of six different categories indicating degrees of content-relation between pages are:

- Identical (comparing a page to itself): 100% content-match.

  1. http://www.google.com/
  2. http://www.yahoo.com/
  3. http://www.doc.ic.ac.uk/
  4. http://www.juventus.it/
  5. http://www.kidscomjr.com/home_flash.html
  6. http://www.cinemorgue.com/
  7. http://www.io.com/~sjohn/plots.htm
  8. http://www.greendesigns.com/
  9. http://www.naspl.org/
  10. http://www.genomenewsnetwork.org/

- 6 key-words query in Google: similar content of pages.

  1. http://www.greenplastic.com/lyrics/rh_songs/creep.php
     http://www.followmearound.com/lyrics/creep.html
  2. http://plato.stanford.edu/entries/hume/
     http://oregonstate.edu/instruct/phl302/philosophers/hume.html
  3. http://www.pink-floyd-lyrics.com/html/one-of-my-turns-wall-lyrics.html
     http://www.azlyrics.com/lyrics/pinkfloyd/oneofmyturns.html
  4. http://books.guardian.co.uk/reviews/generalfiction/0,6121,977505,00.html

http://mostlyfiction.com/world/ali.htm
5. http://www.nietzsche.ru/english/biography.php3
   http://www.kahlil.org/nietzschebio.html
6. http://www.musicomh.com/albums/norah-jones.htm
   http://www.dvdtalk.com/reviews/read.php?ID=6819
7. http://www.doc.ic.ac.uk/teaching/postgraduate/computing_sci
   ence/index.html
   http://www.doc.ic.ac.uk/teaching/postgraduate/mac/
8. http://www.thoracic.org/criticalcare/ccpac/answers/ccpac1101.asp
   http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=29056
9. http://westerncanon.com/cgibin/lecture/Weberhall/cas/36.html
   http://glory.gc.maricopa.edu/~kfurlong/101syllabus.htm
10. http://www.einstein-website.de/biography-e.htm
    http://www.nobel.se/physics/laureates/1921/einstein-bio.html

- 4 key-words query in Google: a high content-based match between pages.

1. http://plato.stanford.edu/entries/hume/
   http://www.utm.edu/research/iep/h/humelife.htm
2. http://www.lyricstime.com/lyrics/16875.html
   http://www.greenplastic.com/lyrics/rh_songs/anyonecanplayguitar.php
3. http://water.usgs.gov/nrp/proj.bib/kharaka.html
   http://www.bgu.ac.il/BIDR/school/syllabe.htm
4. http://jvi.asm.org/cgi/content/full/77/8/4528
   http://www.panspermia.org/replies3.htm
5. http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&
   db=PubMed&list_uids=10024177&dopt=Abstract
   http://web.mit.edu/solab/www/Documents/SoLab_documents.html
6. http://www.math.utah.edu/~cherk/mathjokes.html
   http://www.psych.upenn.edu/humor.html
7. http://www.wordiq.com/definition/List_of_legal_topics
   http://theologytoday.ptsem.edu/oct1995/v52-3-article1.htm
8. http://web.cocc.edu/lminorevans/Freud_week2_214.htm
   http://www.pitt.edu/~frieze/per2set3.htm
9. http://blog.fastcompany.com/archives/2003/09/04/quantifying_cr
   eativity.html
   http://www.synnovation.co.za/Serv.htm
10. http://www.lyricsfreak.com/e/eric-clapton/51435.html
    http://www.lyricsdir.com/e/eric-clapton/after-midnight.php

- 2 key-words query in Google: some degree of relation between pages.

1. http://www.lisawestbergpeters.com/lessons/shake1.html
   http://www.columbiariverpeace.org/hoodstock/lineup.htm
2. http://www.guinnessworldrecords.com/index.asp?id=47603
   http://www.findarticles.com/p/articles/mi_m1134/is_2_110/ai_71317741

3. http://www.genetictestlab.com/
   http://web.mit.edu/esgbio/www/dogma/history2.html
4. http://mind.ucsc.edu/dreams/
   http://sqab.psychology.org/
5. http://www.core77.com/resource/plastique/rot.html
   http://www.tech.oru.se/cad/varkon/v_man/f158.htm
6. http://www.medicalconsumerguide.com/primary_care/heart_vascular_d
   isorders/high_blood_pressure.html
   http://www.holistic-online.com/Remedies/Heart/hypert_her
   b-hypretension.htm
7. http://encyclopedia.thefreedictionary.com/penis%20envy
   http://www.shef.ac.uk/~psysc/psychoanalytic-studies/msg00296.html
8. http://william-king.www.drexel.edu/top/eco/game/game.html
   http://www.ics.uci.edu/~eppstein/cgt/
9. http://www.iangv.org/jaytech/
   http://www.ong.com/
10. http://www.triangle.co.uk/iss/
    http://www.tandf.co.uk/journals/titles/00380415.asp

- 1 key-word query in Google: low content match between pages.

1. http://www.alzheimers.org/
   http://www.amnh.org/exhibitions/epidemic/
2. http://www.health.state.nd.us/ndhd/environ/ee/rad/radon/
   http://www.epa.gov/radon/
3. http://www.cnn.com/
   http://cim.pennnet.com/home.cfm
4. http://www.acsm.org/
   http://www.med.upenn.edu/
5. http://www.mnsu.edu/emuseum/information/biography/
   http://www.number-10.gov.uk/output/page4.asp
6. http://zygote.swarthmore.edu/fert5.html
   http://www.ingenta.com/journals/browse/klu/jomm
7. http://cain.ulst.ac.uk/
   http://www.colorado.edu/conflict/
8. http://www.nmsis.org/
   http://easyweb.easynet.co.uk/simplepsych/antipsych.html
9. http://www.paulsworld.co.uk/beckham/db_main.htm
   http://www.askmen.com/men/sports/30_david_beckham.html
10. http://www.philosophypages.com/
    http://eserver.org/philosophy/

- Randomly chosen pages: thematically unrelated pages.

1. http://www.football.com/
   http://www.infoplease.com/people.html

2. http://www.geocities.com/SunsetStrip/Club/9542/woody.html
   http://www.alaskariveradventures.com/
3. http://www.bushwatch.com/
   http://www.limoneira.com/sales.html
4. http://www.thewaterpage.com/
   http://www.census.gov/ipc/www/idbnew.html
5. http://www.number-10.gov.uk/output/page4.asp
   http://www.dmgi.com/bananas.html
6. http://www.export.gov.il/eng/
   http://pure-essence.net/watermelon/
7. http://message-bottle.warnerbros.com/
   http://www.stonecompany.com/
8. http://www.trueorigin.org/
   http://msnbc.msn.com/id/5549064/
9. http://www.msobczak.com/prog/yami/
   http://www.balloongatineau.com/
10. http://shitin.free.fr/
    http://www.dgp.toronto.edu/people/stam/reality/Talks/index.html

A.2 Design Diagrams

The UML class diagrams below (figures A.1 and A.2) present the Object-Oriented design of the Pagerank-Agent, the main component of the distributed system developed.

A full documentation (javadoc) of the system developed is available online:
www.doc.ic.ac.uk/∼sm1603/project/doc/

The Java source-code and all scripts used for running the system are also available online:
www.doc.ic.ac.uk/∼sm1603/project/src/

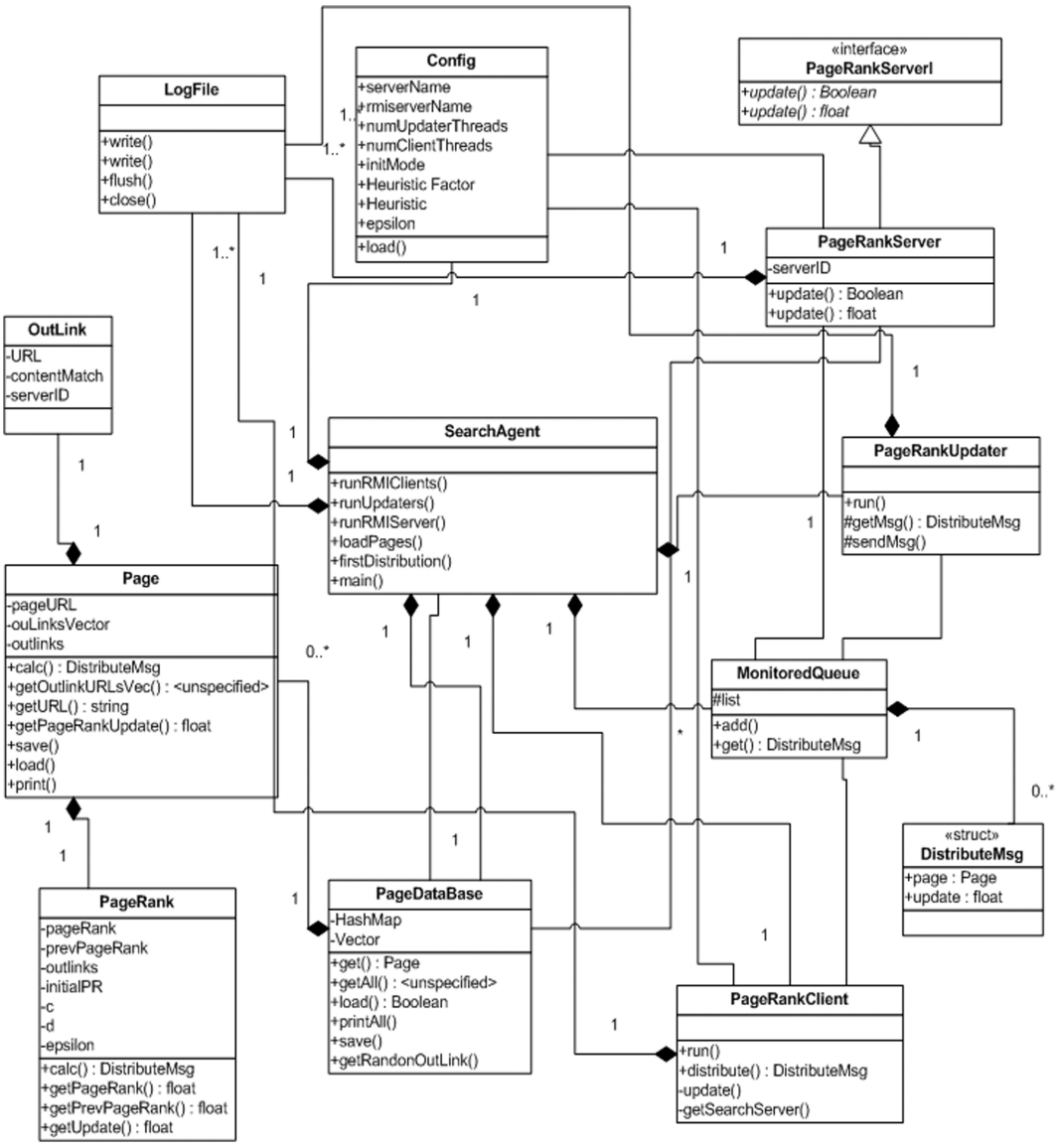

Figure A.1: Link-Based Pagerank Agent's main classes: UML diagram

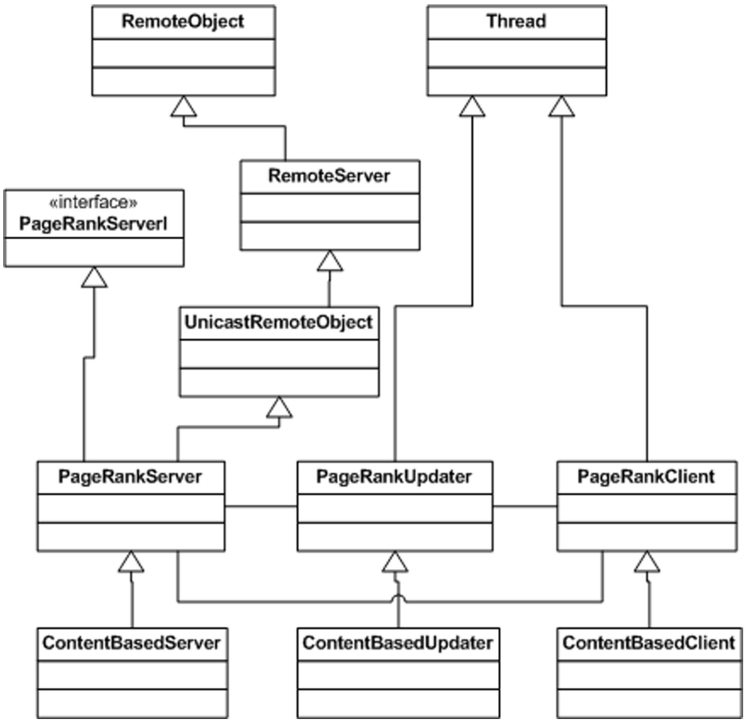

Figure A.2: Link-Based and Content-Sensitive main classes: UML inheritance diagram

Bibliography

[BP98] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. Computer Science Department, Stanford University, Stanford, 1998.

[CM69] D. Chazan and W. Miranker. In *Linear Algebra and its Applications, 199-222*. 1969.

[Cra] Website specific processosrs for html information extraction. In *http://www-2.cs.cmu.edu/ rcm/websphinx*. School of Computer Sciences, Carnegie Mellon University.

[D-L] D-lib magazine. In *http://www.dlib.org/*.

[EFT] Efactory web site. In *http://pr.efactory.de/*.

[Goo] Google search engine. In *http://www.google.com/*.

[Hav02] Taher H. Haveliwala. Topix-sensitive pagerank. Computer Science Department, Stanford University, Stanford, 2002.

[KBS00] Farzin Maghoul Krishna Bharat and Raymie Stata. The term vector database: fast access to indexing terms for webpages. In *Computer Networks*. In Proceedings of the 9th International World Wide Web Conference, Amsterdam, Netherlands, 2000.

[KSB03] Simha Sethumadhavan Kathikeyan Sankaralingam and James C. Browne. Distributed pagerank for p2p systems. Department of Computer Sciences, The University of Texas at Austin, 2003.

[MKM00] Behrang Mohit Mark Kantrowitz and Vibhu Mittal. Stemming and its effects on tfidf ranking. In *Annual ACM Conference on Research and Development in Information Retrival, pages 357-359*. ACM Press, New York, NY, USA, 2000.

[ODP] Open directory project. In *http://dmoz.org/*.

[Par] Htmlparser. In *http://htmlparser.sourcefroge.net/*. Sourcefoge Project.

[PB98] L. Page and S. Brin. The pagerank citation ranking: Bringing order o the web. Computer Science Department, Stanford University, Stanford, 1998.

[Por80] M. F. Porter. An algorithm for suffix stripping. In *Program, 14(3), 130-137*. 1980.

[RD02] Matt Richardson and Pedro Domingos. The intelligent surfer: Probabilistic combination of link and content information in pagerank. In *Advances in Neural Information Processing Systems 14*. MIT Press, 2002.

[SEO] Search engine optimization chat. In *http://www.seochat.com/*.

[SS95] Amit Singhal and Gerard Salton. Automatic text browsing using vector space model. Department of Computer Science, Cornell University, Ithaca, NY, 1995.

[SSW03] GuangWen Yang ShuMing Shi, Jin Yu and DingXing Wang. Distributed page ranking in structured p2p networks. Department of Computer Science and Technology, Tsinghula University, Beijing, P.R China, 2003.

[Str97] John C. Strikwerday. A convergence theorem for chaotic asynchronous relaxation. Computer Sciences Department University of Wisconsin-Madison, 1997.

[Top] Top-ten guarenteed. In *http://www.1stspot.com/*.

[YDi] Yahoo directory. In *http://dir.yahoo.com/*.